

# Linguagem de Programação Orientada a Objetos I

Introdução a Programação Orientada a Objetos  
Prof. Tales Bitelo Viegas

<https://fb.com/ProfessorTalesViegas>

# Introdução à POO

## ► Modelos

- São representações gráficas simplificadas de objetos, pessoas, itens, tarefas, processos, conceitos, idéias, usados comumente por pessoas no dia a dia, independente do uso de computadores
- Exemplo 1: pessoa como paciente de uma clínica médica
  - Para modelar o paciente seria necessário: representar nome, idade, altura, peso, histórico de consultas
  - Operações: verificarObesidade, adicionarInformaçãoAoHistórico

# Introdução à POO

- Exemplo 2: pessoa como contato comercial
  - Para modelar o contato seria necessário: nome, telefone, cargo, empresa
  - Operações: mostrarTelefone, consultarEmpresa

# Introdução à POO

- ▶ Programação OO (POO)
  - Paradigma de programação de computadores onde se usam classes e objetos, criados a partir de *modelos*, para representar e processar dados usando programas de computadores

# Introdução à POO

- ▶ Exemplos: Registro Acadêmico de um aluno
  - Modelo do Registro Acadêmico do Aluno

## Aluno

nome: String

cgu: int

dataNascimento: String

definirNome(novoNome: String): void

buscarNome(): String

calcularIdade(): int

# Introdução à POO

- ▶ Exemplos: Uma lâmpada
- ▶ Modelo da Lâmpada

## Lâmpada

estadoLampada: boolean

acender(): void

apagar(): void

mostrarEstado(): boolean

# Introdução à POO

- ▶ Exemplos: Uma conta bancária
  - Modelo da Conta

## ContaBancaria

nomeCorrentista: String

saldo: int

contaEspecial: boolean

abrirConta(nome: String, valorDeposito: float, especial: boolean): void

depositar(valor: float): void

sacar(valor: float): void

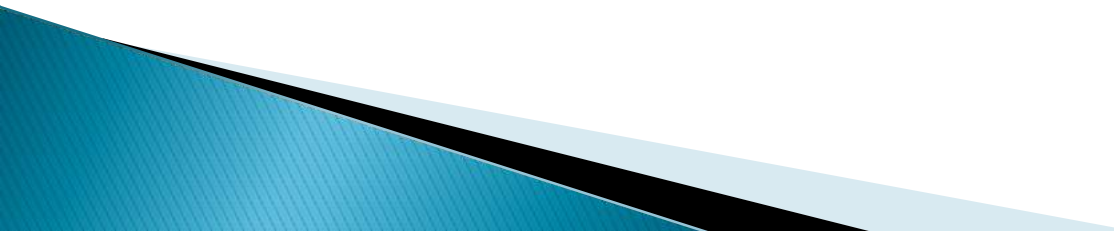
mostrarDados(): void

# Conceitos de OO – Visão Geral

- ▶ Programadores que utilizam o paradigma orientado a *objetos* criam e usam objetos a partir de *classes*, que são relacionadas diretamente com modelos



# Conceitos de OO – Classes

- ▶ *Classes* são estruturas das linguagens de programação OO para conter, para determinado modelo, os dados que devem ser representados e as operações que devem ser efetuadas com estes dados
  - ▶ Cada *Classe* deve ter um nome que seja facilmente associado ao modelo que a classe representa
  - ▶ Representam o modelo
- 

# Classe em Java

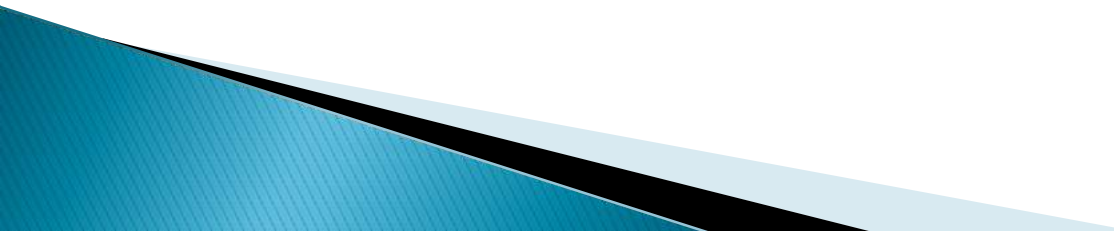
```
class Aluno{
```

```
}
```



Definição da Classe

# Conceitos de OO – Atributos

- ▶ Os dados contidos em uma classe são conhecidos como *campos* ou *atributos* daquela classe
  - ▶ Este campo deve ter um nome e tipo, que será ou um tipo de dado nativo da linguagem ou uma classe existente na linguagem ou definida pelo programador
  - ▶ Se a classe é usada para que várias instâncias sejam criadas a partir dela, cada uma dessas instâncias terá um conjunto dos campos definidos na classe
- 

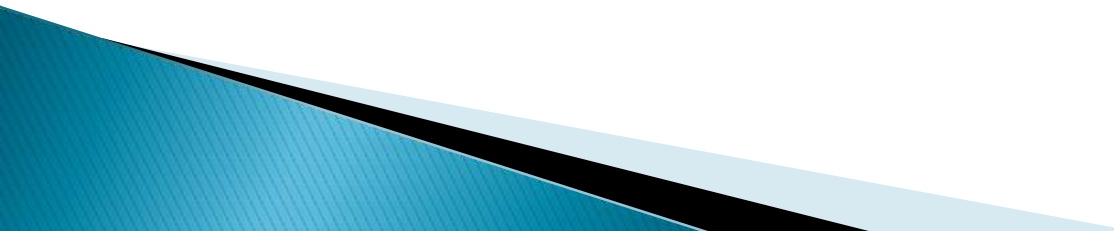
# Atributos de uma Classe

```
public class Aluno{  
    String nome;  
    int cgu;  
    String dataNascimento;  
}
```

← Definição da Classe

← Atributos

# Conceitos de OO – Métodos

- ▶ As operações contidas em uma classe são chamadas de métodos dessa classe
  - ▶ Métodos são geralmente chamados ou executados explicitamente a partir de outros trechos de código na classe que o contém ou a partir de outras classes
  - ▶ Métodos podem opcionalmente receber *argumentos para métodos*
  - ▶ Métodos podem opcionalmente retornar valores ou instâncias de classes
- 

# Classes

```
public class Aluno{
```



Definição da Classe

```
    String nome;
```

```
    int cgu;
```

```
    String dataNascimento;
```



Atributos

```
    public void definirNome(String n) {
```

```
        nome = n;
```

```
    }
```

```
    public String buscarNome() {
```

```
        return nome;
```

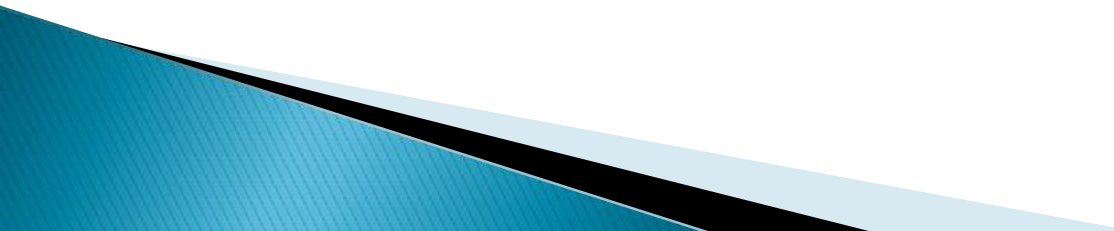
```
    }
```

```
}
```



Métodos

# Conceitos de OO – Encapsulamento

- ▶ O mecanismo de encapsulamento é uma forma de restringir o acesso ao comportamento interno de um objeto
  - ▶ Um objeto que necessite de colaboração de outro objeto para realizar uma tarefa simplesmente envia uma mensagem a este último
  - ▶ Dados e comportamento especificados num mesmo módulo
- 

# Conceitos de OO – Encapsulamento

- ▶ Abstração: esconder os detalhes de funcionamento de um objeto
- ▶ Separam utilização e implementação
  - promovem reutilização
  - implementação independente do contexto de uso

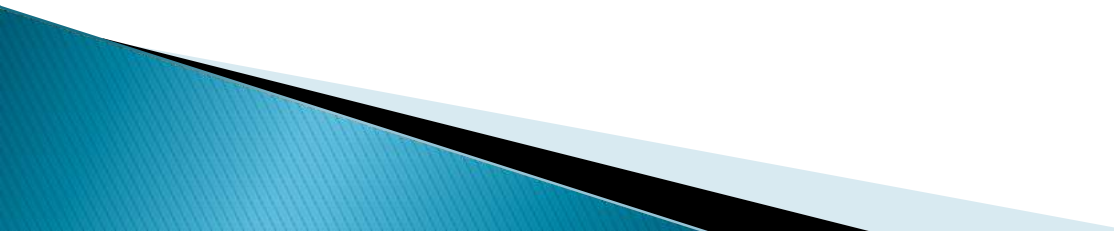


# Atributos Públicos x Privados

## ▶ Atributos Públicos

- Atributos podem ser acessados e modificados a partir de qualquer classe
- Não propicia o encapsulamento de dados

## ▶ Atributos Privados

- Atributos só podem ser acessados e modificados a partir de métodos da própria classe que a pertencem
  - Propicia o encapsulamento de dados
- 

# Classes

```
public class Aluno{  
    public String nome;  
    public int cgu;  
    public String dataNascimento;
```



Atributos Públicos

```
    public void definirNome(String n){  
        nome = n;  
    }  
    public String buscarNome(){  
        return nome;  
    }  
}
```



Método Público

# Classes

```
public class Aluno{
```

```
    private String nome;
```

```
    private int cgu;
```

```
    private String dataNascimento;
```

```
    public void definirNome(String n){
```

```
        nome = n;
```

```
    }
```

```
    public String buscarNome(){
```

```
        return nome;
```

```
    }
```

```
}
```



Atributos Privados



Método Público

# Métodos

- ▶ Declaração de um método
  - <acesso><tipo><nome>(<parametros>)
- ▶ Assinatura de um método
  - Nome + tipos e números de parâmetros (independente do nome das variáveis)

# Métodos

- ▶ Os métodos podem ser:
  - Públicos: podem ser acessados a partir de qualquer classe
  - Privados: só podem ser acessados a partir de métodos da própria classe que a pertencem (proteger métodos que não interessam a outras classes – métodos de implementação)
- ▶ Exemplos
  - `void metodo1()` // não têm parâmetro e não retorna nada
  - `public int metodo2()` // retorna um dado inteiro
  - `public int metodo2(int valor)` // passa um inteiro como parâmetro

# Métodos

- ▶ Declaração: cabeçalho (interface)
  - valor de retorno
  - nome
  - lista de parâmetros
- ▶ Definição: corpo (código do método)

```
public void setSaldo (double valor){  
    saldo = valor;  
}  
public int getSaldo(){  
    return (saldo);  
}
```

# Abributos e Métodos


- ▶ O que vai ser privado e público?
- ▶ Regra geral:

## **Classe**

**Atributos: privados**

**Métodos: privados (se forem usados apenas dentro da classe) ou públicos (se forem usados fora da classe)**

# Métodos Acessores (get, set, is)

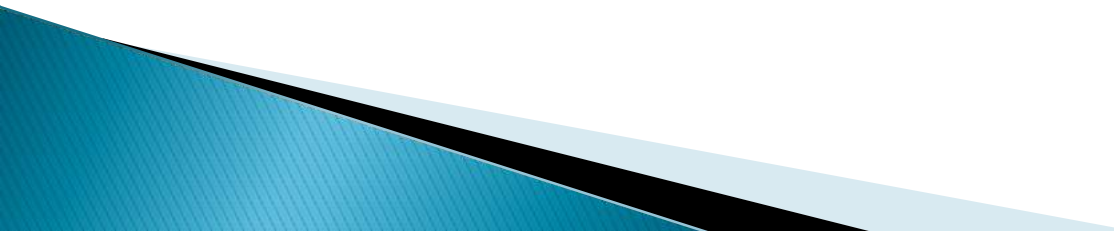
- ▶ Permitem uma forma de acessar os atributos privados de uma classe
  - ▶ Métodos get são métodos que permitem retornar o valor de um atributo
  - ▶ Métodos set são métodos que permitem definir o valor de um atributo
  - ▶ Métodos is são usados para retornar o valor de um atributo boolean
- 



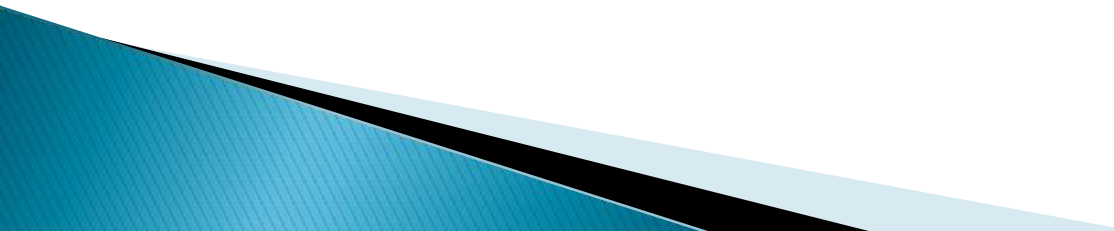
# Métodos Acessores (get, set, is)

```
public class Aluno {  
    private String nome;  
    private int cgu;  
    private String dataNascimento;  
    public String getNome() {  
        return nome;  
    }  
    public void setNome(String nome) {  
        this.nome = nome;  
    }  
    public int getCgu() {  
        return cgu;  
    }  
    public void setCgu(int cgu) {  
        this.cgu = cgu;  
    }  
    public String getDataNascimento() {  
        return dataNascimento;  
    }  
    public void setDataNascimento(String dataNascimento) {  
        this.dataNascimento = dataNascimento;  
    }  
}
```

# Conceitos de OO – Métodos

- ▶ Mensagens = chamada de métodos
  - ▶ Para que uma operação seja executada, deve haver um estímulo enviado a esse objeto
  - ▶ Os objetos de um sistema trocam mensagens através dos métodos
- 

# Conceitos de OO – Objetos

- ▶ Para a representação de dados específicos usando classes será necessária a criação de *objetos* ou *instâncias* desta classe
  - ▶ Um *objeto* ou *instância* é a materialização de uma classe (usada para representar dados e executar ações)
  - ▶ Para que os objetos ou instâncias possam ser manipulados, é necessária a criação de *referências* a estes objetos, que são basicamente variáveis do “tipo” classe
- 

# Objetos

- ▶ Declaração de variável
  - Associa um nome (de objeto) a uma classe
  - Exemplo: `Conta conta1;`
- ▶ Instanciação
  - Criação/inicialização de um objeto
  - Comando **new**
  - Exemplo: `conta1 = new Conta();`
- ▶ Declaração + Instanciação
  - Exemplo: `Conta conta1 = new Conta();`

# Objetos

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

```
        Aluno aluno1; ← Declarando
```

```
        aluno1 = new Aluno(); ← Instanciando
```

```
        Aluno aluno2 = new Aluno(); ← Declarando  
                                         e Instanciando
```

```
    }
```

```
}
```

# Objetos

```
public class Main {  
    public static void main(String[] args) {  
  
        Aluno aluno1 = new Aluno();  
        Aluno aluno2 = new Aluno();  
  
        int idadeAluno1, idadeAluno2;  
        idadeAluno1 = aluno1.calculaIdade();  
        idadeAluno2 = aluno1.calculaIdade();  
    }  
}
```




Chamando  
métodos

# Objetos

```
public class Main{  
    public static void main(String[] args) {  
  
        Aluno aluno1 = new Aluno();  
        Aluno aluno2 = new Aluno();  
  
        aluno1.nome = "Tales";  
    }  
}
```

Acesso  
Ilegal  
(sendo o atributo privado)



**Pergunta:** e se o atributo for público ?

# Objetos


- ▶ **Objetos podem ser:**
  - **Copiados:** fazer uma cópia de cada campo de um objeto em outro objeto
    - Exemplo: `aluno1.nome = aluno2.nome; // deverão ser públicos`
  - **Atribuídos:** fazer com que a referência de um objeto seja substituída pela referência de outro objeto
    - Exemplo: `aluno1 = aluno2;`
    - Diferente de tipos primitivos que copiam valores



# Objetos

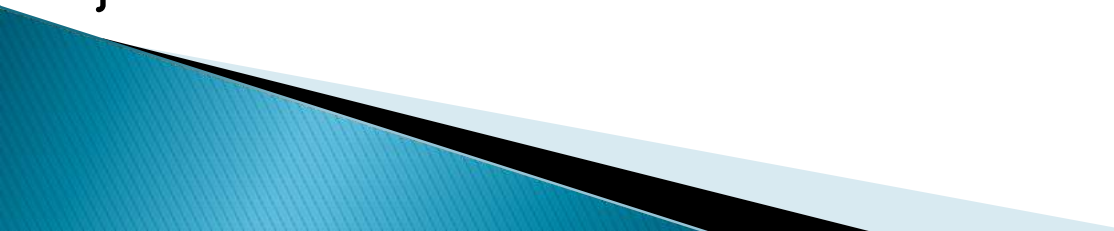
- ▶ Objetos podem ser:
  - Usados como parâmetros/passados como argumentos/devolvidos como resultados
    - Exemplo: `Aluno relacionaAluno(Aluno a) { ... }`

# Atribuição de Objetos

```
public class Main{  
    public static void main(String[] args) {  
  
        Aluno aluno1 = new Aluno();  
        Aluno aluno2 = new Aluno();  
        aluno1 = aluno2;   
        // aluno1 e aluno2 apontarão para o mesmo  
        objeto  
        // uva e pera são handles ou referencias  
    }  
}
```

# Atribuição de Objetos

```
public class Main{  
    public static void main(String[] args) {  
        Aluno aluno1 = new Aluno();  
        System.out.println("Id obj aluno1 = " + aluno1);  
        Aluno aluno2 = new Aluno();  
        System.out.println("Id obj aluno2 = " + aluno2);  
        aluno1 = aluno2;  
        // aluno1 e aluno2 apontarão para o mesmo objeto  
        // uva e pera são handles ou referencias  
        System.out.println("Id obj aluno1 = " + aluno1);  
    }  
}
```



# Atribuição de Campos

```
public class Main{  
    public static void main(String[] args) {  
        Aluno aluno1 = new Aluno();  
        Aluno aluno2 = new Aluno();  
        aluno1.nome = aluno2.nome;  
        aluno1.cgu = aluno2.cgu;  
    }  
}
```

Os atributos deverão ser públicos !

**Pergunta:** isto é bom para o encapsulamento ?

# Colocando em prática

- ▶ Exercícios
  - Ver no Moodle.