Criação de Objetos e Acesso à Métodos

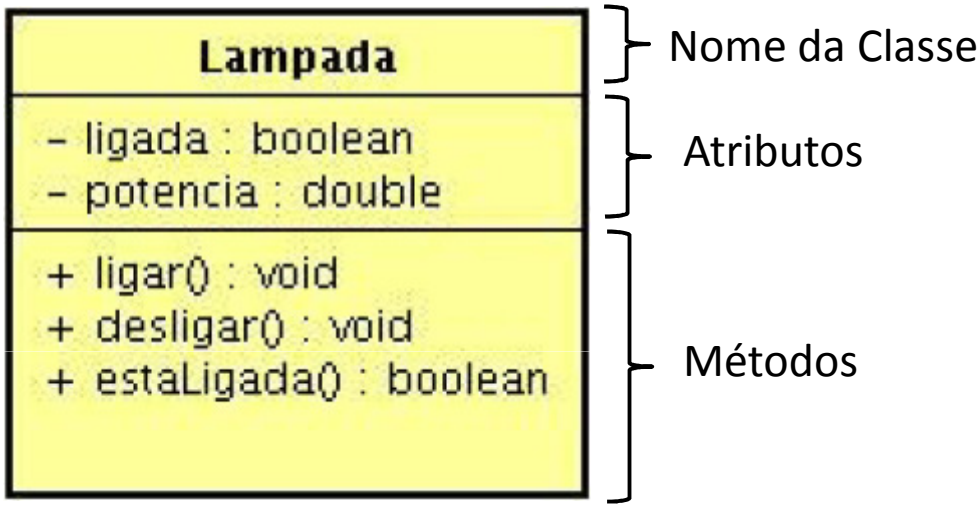
Prof. Bruno Gomes

bruno.gomes@ifrn.edu.br

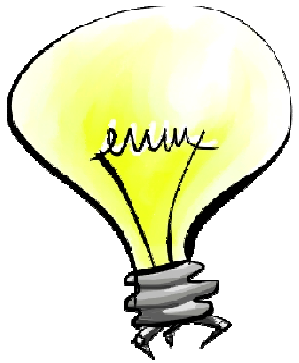
Programação Orientada a Objetos

Classe

- Classe:



- *Classe Lampada*
  - *Atributos*
    - *potencia (double), ligada (boolean)*
  - *Operações*
    - *ligar, desligar, estaLigada*



Métodos em Java

- Sintaxe:

```
[<modificadores_método>] <tipo_retorno> <nome_método> ([<parametros>]){

        // Corpo do Método

}
```

[ ] = Opcionais
< > = Identificadores e palavras reservadas

Métodos em Java

- Passagem de parâmetros:
  - Deve ser informados o tipo e identificador dos parâmetros
  - Funciona no método como uma variável normal
  - Passam o valor do identificador

```
void sacar(double valorSacado){
        valor-=valorSacado;
}

void depositar(double valorDepositado){
        valor+=valorDepositado;
}
```

Métodos em Java (Lembrando)

- Usamos o operador "." (ponto) para acessar um método
  - Sintaxe:
    - objeto.método();
  - Executa método em objeto
    - Objeto deve existir
  - A variável deve referenciar objeto válido
    - Se referenciar null ocorre erro
  - Exemplos:
    - obj1.nomeMetodo();
    - obj1.nomeMetodo(arg1, arg2);
    - (new NomeClasse()).nomeMetodo();
    - obj1.nomeAtributo;

Corpo do Método

- Corpo do método:
  - Implementa as operações do método
  - Fica entre chaves ({})
  - Variáveis podem ser criadas
    - Ela é dita local
    - Não é pré-inicializada
    - Só existe enquanto o método está em execução

Método

- É possível que uma Classe possua 2 métodos com o mesmo nome?

Sim, é possível, mas devem ter parâmetros diferentes (quantidade e tipo)!!!
O nome que se dá a isso é **sobrecarga** ou **clonagem**!

```
double calcularMedia(double nota1, double nota2){
return (nota1+nota2)/2;
}

double calcularMedia(double nota1, double nota2, int peso1, int peso2){
return (nota1*peso1+nota2*peso2)/peso1+peso2;
}
```

Atributos

- São as variáveis de instância
  - Fazem parte de cada objeto (instância)
- Declarada fora dos métodos
- "Vivem" enquanto o objeto "viver“
- Obs: Todo objeto possui um identificador chamado **this**, que é uma referência para o próprio objeto.

Atributos

[<modificadores_atributo>] <tipo_atributo> <nome_atributo> [= valor_inicial];

[ ] = Opcionais

< > = Identificadores e palavras reservadas

```
public boolean estadoLampada = false;
Double valor;
String marca = “fluorescente”;
```

this

- Todo objeto possui um atributo que é uma referência a ele mesmo
  - Usado para acesso a membros do próprio objeto
  - **this.membro**
  - Evita conflito Com parâmetros de métodos, por exemplo

```
class NomeClasse {
        int x, y;
        public void mover(int x,int y){
                this.x = x;
                this.y = y;
        }
}
```

Classe Completa

```
public class Lampada {

public boolean estadoLampada;

        public Lampada(){
                estadoLampada = false;
        }

        public void acenderLampada() {
                estadoLampada = true;
        }

        public void apagarLampada() {
                estadoLampada = false;
        }

        public boolean verEstadoLampada() {
                return estadoLampada;
        }
}
```

Atributos

Construtor

Métodos

Construtor

- Mesmo nome da classe
- Não possui retorno
- Uma classe pode conter vários construtores
  - Diferença na **quantidade** e **tipo** dos parâmetros
- Construtor padrão é fornecido
  - Se não houver pelo menos um definido
  - Não possui parâmetros
- É chamado na execução do **new**

Construtor

Classe com 2 construtores

```
public class Pessoa {
        String nome;
        int rg, cpf;

        public Pessoa(){
                nome="";
                rg=0;
                cpf=0;
        }

        public Pessoa(String nome, int rg, int cpf) {
                this.nome = nome;
                this.rg = rg;
                this.cpf = cpf;
        }
}
```

Construtor

- Criando objeto de uma classe com 2 construtores:

```
Pessoa obj = new Pessoa();
Pessoa obj2 = new Pessoa("Bruno", 1234, 1234567890);
```

Exercício 1

- Crie uma classe Calculadora, onde a mesma terá 4 métodos: **somar, subtrair, dividir** e **multiplicar**.
  - Todos os métodos recebem 2 valores reais como **parâmetros**, e **retornam** o resultado da operação
- Crie outra classe, com o método **main**, para testar a Calculadora.
  - Crie um objeto calculadora, e realize as 4 operações acessando os métodos oferecidas por ela

Exercício 2

- Crie uma classe **Pessoa**. Nela terá os atributos **nome**, **idade**, **cpf**.
  - Crie um **construtor parametrizado** inicializando todas as variáveis com os valores recebidos dos parâmetros.
  - Crie um **construtor default** (Inicializando as variáveis da classe com valores padrões).
  - Crie um **método** para **receber os 3 valores** dos atributos da classe Pessoa e alterá-los.
- Crie outra classe, com o método **main**, para testar a classe Pessoa:
  - Nela, crie **2 objetos** da classe Pessoa. Um dos objetos criados deve **inicializar as variáveis pelo construtor**. O segundo objeto deve usar o **construtor default** para criar o objeto**,** e mudar os valores de Pessoa acessando o método de alterar

Trabalho

- Redigir um relatório sobre API (*Application Programming Interface*). Nele:
  - Explicar o que é API e para que serve
  - Explicar como acessar a API Java (site, onde clicar, versão)
  - Explicar onde estão localizadas as informações dentro da API

Trabalho – Cont.

- No relatório, mostrar como utilizar as classes:
  - **Math** para cálculos matemáticos
  - **Calendar** para manipulação de datas
- Citar pelo menos 5 operações (métodos) realizadas por cada Classe, mostrar exemplos em Java.

- **Obs.: Todas as informações do relatório devem ser tiradas diretamente da API do Java**

Trabalho – Cont.

- Entregar relatório impresso, com nome e matrícula
- **Valor:** 1,0 ponto
- **Entrega:** 31/03/10