Introdução a Orientação a Objetos

Prof. Ms. Peter Jandl Junior
Prof. Ms. Télvio Orru
Prof. Nathan Silva

J12B

Linguagem de Programação Orientada a Objetos

Ciência da Computação - UNIP – Jundiaí

Programação orientada a objetos
introdução

Um novo paradigma de programação de computadores

POO :: introdução

- Esta apresentação contém os conceitos introdutórios da Programação Orientada a Objetos – POO, um novo paradigma de programação de computadores cujas características são as melhores conhecidas para a análise, o projeto e o desenvolvimento de sistemas.
- Mostra ainda as características básicas de uma linguagem de programação em relação a orientação a objetos.

Motivações

Princípios

Conceitos Chave

Primeiro Objeto

POO:Motivações

Desenvolvimento de software clássico

- Resolução de problemas baseadas na divisão em partes mais simples
  - Funções
  - Módulos e bibliotecas
  - Sub-programas.
- Técnicas de programação voltadas para estruturação da solução em tais partes.
- Linguagens de programação voltadas para o modo de funcionamento do computador.

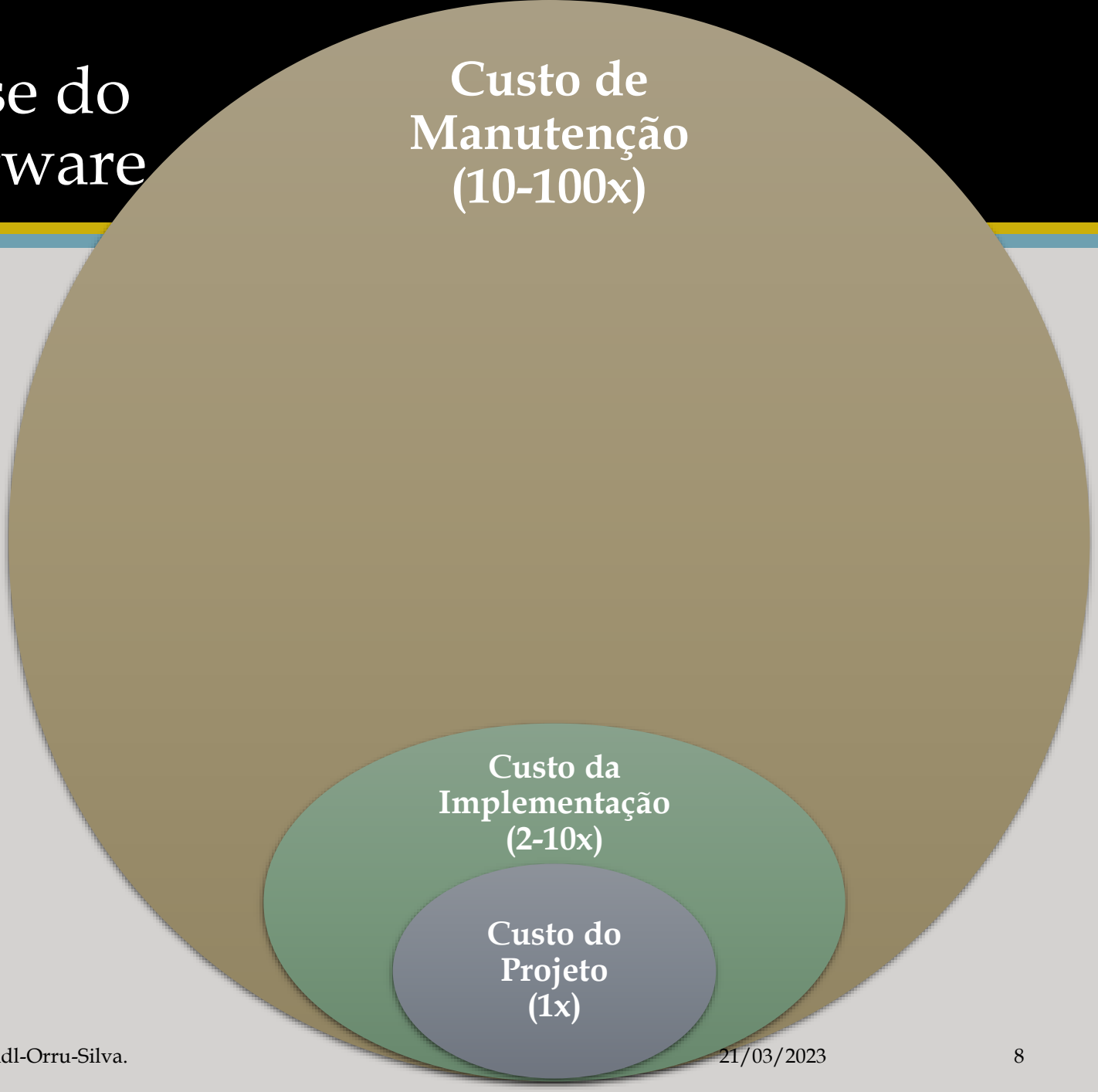
Evolução dos sistemas

- Com a evolução dos computadores e a sofisticação das necessidades das empresas e dos usuários ...
- Software tornou-se maior e mais complexo, exigindo equipes mais numerosas para seu desenvolvimento ...
- Capacidades individuais tornam-se cada vez menos determinantes no resultado produzido pelas equipes.

Crise do software

- Com o aumento da complexidade do software e do tamanho das equipes:
  - Prazos tornam-se cada vez mais difíceis de cumprir!
  - Custos fogem sempre as previsões!
  - Qualidade do software deixa sempre a desejar!

Crise do Software



Crise do software

- Para reverter o cenário da crise:
  - Projetos de software passam a ser encarados como projetos de engenharia (de software).
  - Técnicas formais de análise, projeto e desenvolvimento passam a ser estudadas, testadas e avaliadas.
  - Novas linguagens e ferramentas de programação são criadas para atender as novas necessidades dos desenvolvedores.
  - Surge a **Programação Orientada a Objetos**!

Por que Programação Orientada a Objetos?

- Existem diversas motivações para utilizar a POO:
  - desenvolvimento mais rápido e barato;
  - processo mais simples de *design*; e
  - existência de recursos sofisticados de programação.

Programação Orientada a Objetos

- Por outro lado exige:
  - **mudança de paradigma com relação a interpretação dos problemas.**

Na verdade, é um retorno ao nosso modo natural de encarar o mundo.

Programação Orientada a Objetos

- A POO é centrada no projeto, implementação e uso de objetos.
- Como segundo objetivo pretende-se que tais objetos sejam reusáveis.
- Um terceiro objetivo é que o projeto (i.e., a solução do problema) seja reusável (*design pattern*)!

Programação Orientada a Objetos

- A POO requer assim:
  - o emprego de uma nova técnica de projeto,
  - o uso de novas ferramentas de desenvolvimento e
  - a aplicação de novas bibliotecas nesta tarefa.

POO:Conceitos Essenciais

Conceitos Essenciais da OO

- Uma **classe** é um modelo, um conceito **abstrato** para famílias de objetos. As classes podem se relacionar embora sejam distintas.
- Um **objeto** é um representante real, uma entidade **concreta** de uma classe e pode ser distinguido dos demais objetos de sua classe e de outras.

Conceitos Essenciais da OO

- **Classes=Tipos**: o conceito de classe é semelhante a um tipo.
- **Objetos**: São entidades concretas que pertencem a uma classe, i.e., são de um certo tipo.
- **Mensagens**: São pedidos de ação (acompanhados ou não de dados) que podem ser enviados para objetos.

Orientação a Objetos

- Modelo de programação mais natural do que a programação estruturada/funcional.
- Entende tudo como objetos, associando características (atributos) e capacidades (métodos) pertencentes a certos tipos.

POO:Características

OO::Características

- A OO nos oferece 5 características importantíssimas:
  - Abstração
  - Identidade
  - Encapsulamento
  - Herança
  - Polimorfismo

De fato são duas visões de uma mesmo mecanismo.

Características de Linguagens OO

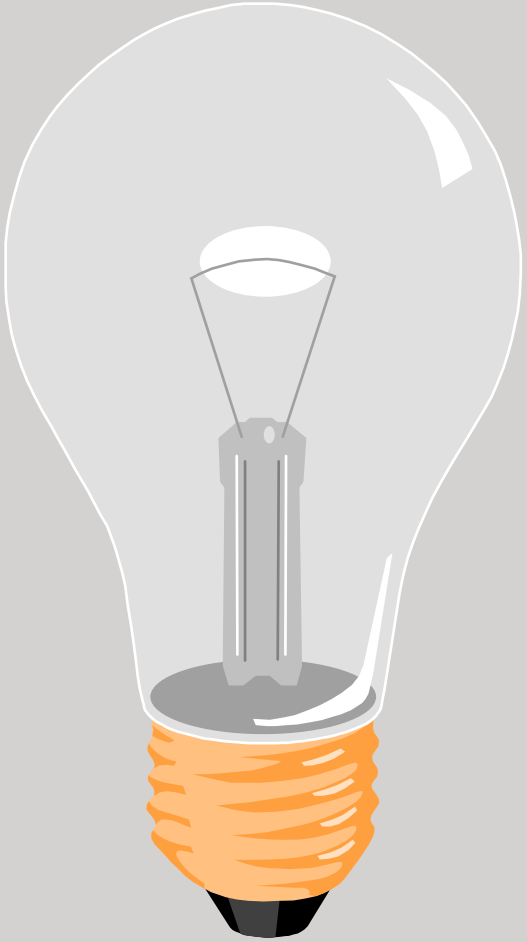
1. Tudo deve ser tratado como um **objeto**.
2. Todo **objeto** é de um tipo específico.
3. Cada **objeto** tem uma "memória", que pode ser composta de outros objetos.
4. Um programa é um **conjunto de objetos** que trocam mensagens.
5. Todos os **objetos** de um mesmo tipo podem receber as mesmas mensagens.

POO com Java

- Java é uma linguagem de programação projetada com a concepção da orientação a objetos.
- Fornece mecanismos claros e diretos para construção de tipos e definição de encapsulamento.
- Permite a criação de novos tipos baseados em herança, além de suportar várias formas de polimorfismo.
- Também inclui autoboxing, tipos genéricos e lambdas.

POO:Um Primeiro Objeto

Um Primeiro Objeto

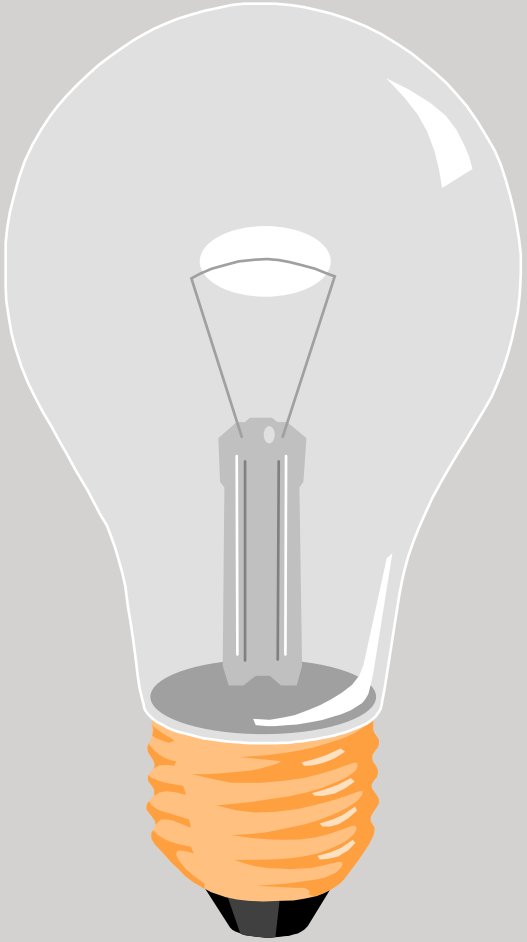


- Uma lâmpada é um **objeto** que possui certas características e comportamentos.
- As ideias mais gerais sobre as lâmpadas determinam um tipo particular de **objeto**, ou o tipo lâmpada.

Abstração

Um Primeiro Objeto

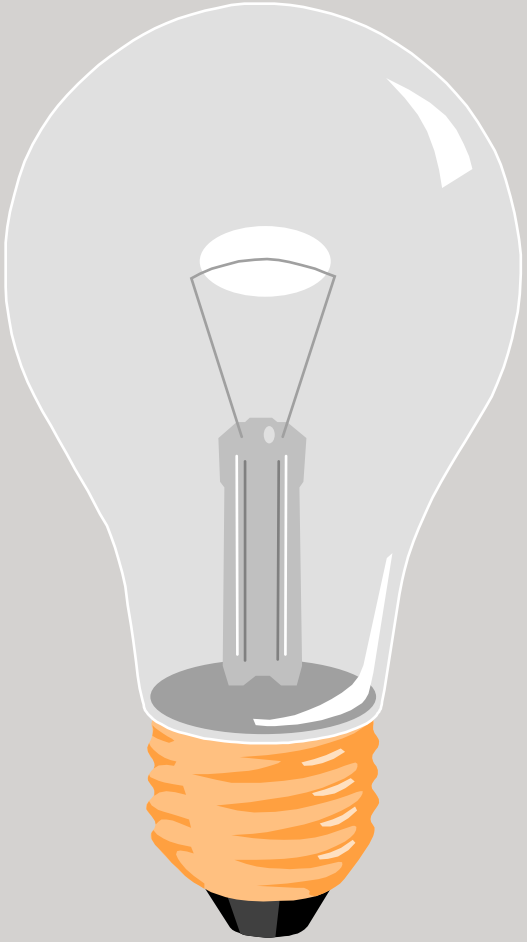
Abstração



- Uma lâmpada em especial tem características bem determinadas embora exiba comportamento semelhante.
- Uma lâmpada pode estar acesa ou apagada, caracterizando dois estados distintos possíveis.

Estados

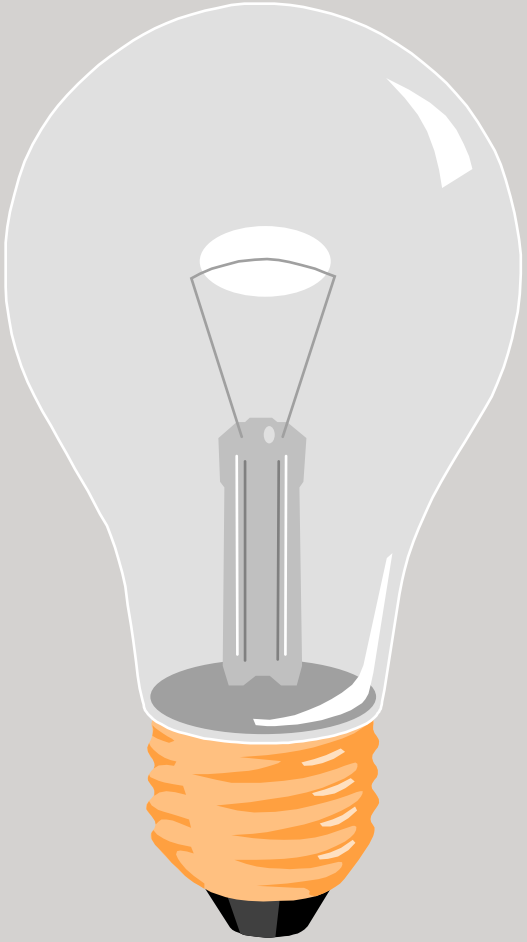
Um Primeiro Objeto



- Uma lâmpada pode ser ligada, o que faz seu estado se tornar acesa.
- Podendo igualmente ser desligada, tornando seu estado apagada.
- Associamos ao objeto lâmpada operações que a colocam num certo estado entre dois possíveis.

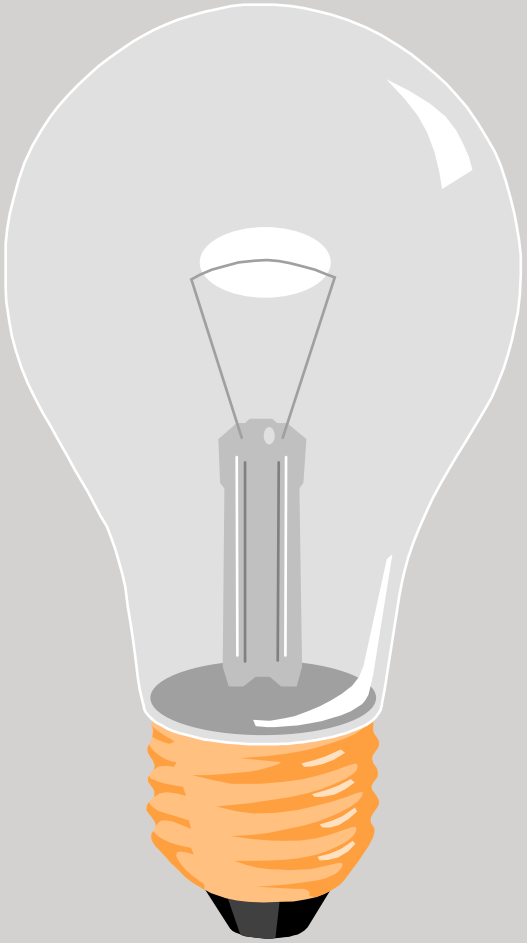
Um Primeiro Objeto

Abstração



- Mesmo não conhecendo os princípios físicos que explicam o funcionamento de uma lâmpada, ela pode ser usada desde que tenhamos acesso às operações que modificam seu estado de funcionamento, ou seja, as operações de:
  - ligar
  - desligar

Um Primeiro Objeto



Conceitualmente:

- Tipo do Objeto: **Lâmpada**
- Interface: ligar()
  desligar()
- Estado Interno: acesa

Capacidades

Estado

Estrutura dos arquivos Java

- Arquivos Java são compostos de:
  - Uma declaração de pacote;
  - Uma ou mais diretivas de importação;
  - Uma ou mais declaração de classes (apenas uma pode ser pública);
  - Uma ou mais declaração de interfaces (apenas uma pode ser pública).

- Os arquivos fonte, de extensão *.java*, **devem** possuir o mesmo nome do elemento público (classe ou interface).

Estrutura de uma classe Java

- Uma classe Java:
  - define um **tipo** de objeto;
  - possui sempre um **especificador de acesso**; e
  - um **nome**.
- Pode possuir:
  - **atributos** (características/estados do tipo)
  - **métodos** (operações/funções do tipo); e
  - **construtores** (operações de criação).

Estrutura de uma classe Java

- A classe *pode* possuir diversos **campos** (atributos ou variáveis internas) para representar seus estados ou características.
- A classe *pode* possuir vários **métodos**, ou seja, funções (operações internas) sobre seus estados.
- A classe também *pode* possuir diversos **construtores** destinados a realizar as operações de construção (criação) de seus objetos.

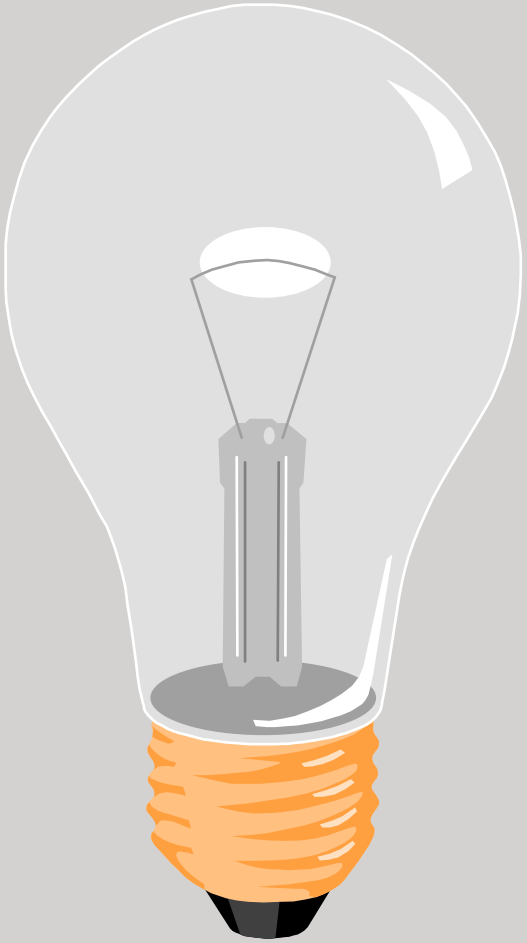
Estrutura de uma classe Java

```
// nível de acesso e nome
acesso class NomeDaClasse {
    // declaração dos campos da classe
    // acesso, tipo e nome
    acesso tipo nomeDoCampo

    // declaração dos métodos da classe
    // acesso, retorno, nome e lista de parâmetros
    acesso tipo nomeDoMétodo (lista_parâmetros) {
    }

    // declaração dos construtores da classe
    // acesso nome (= da classe) e lista de parâmetros
    acesso nomeDaClasse (lista_parâmetros) {
    }
}
```

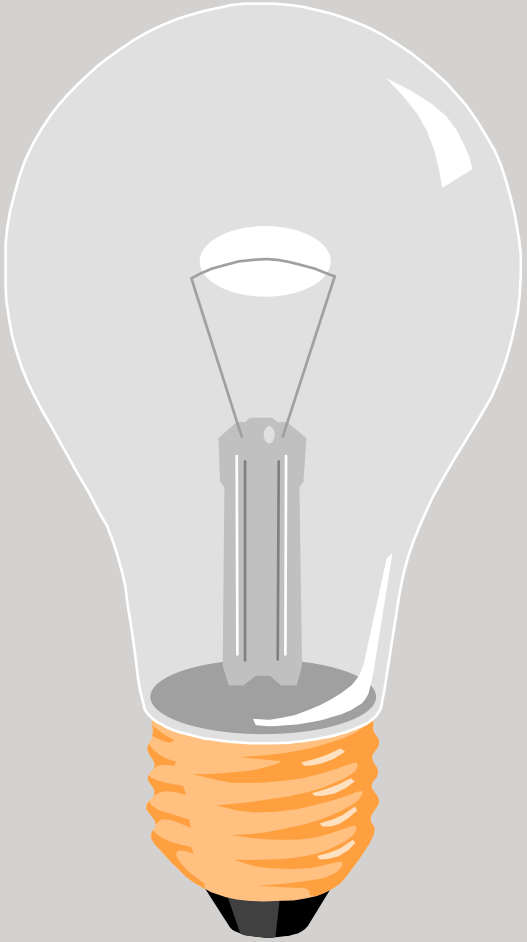
Uma Primeira Classe



Programaticamente em Java:

```
public class Lampada {
    public void ligar() {
            acesa = true;
            System.out.println("Lampada: " +
                                        acesa);
    }
    public void desligar() {
            acesa = false;
            System.out.println("Lampada: " +
                                        acesa);
    }
    private boolean acesa;
}
```

Um Primeiro Objeto

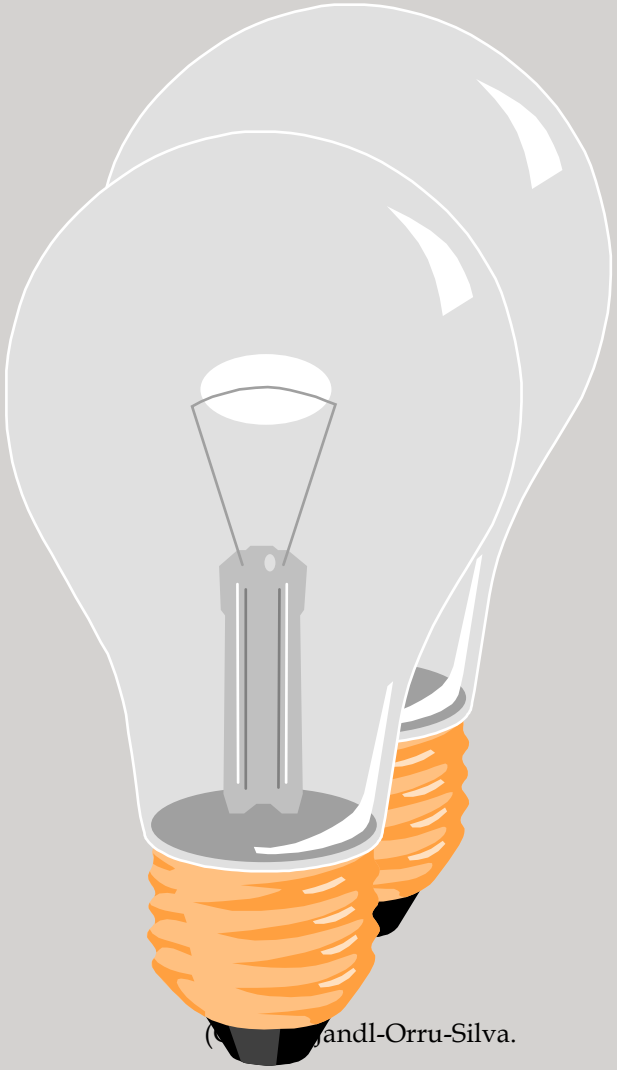


Usando a classe Java:

```
// declarando um objeto
Lampada l;
// instanciando o objeto
l = new Lampada();
// usando o objeto
l.ligar();
l.desligar();
// ao sair de escopo,
// o objeto é removido
```

Identidade

Outros Objetos da mesma Classe



Usando a classe Java:

```
// declarando vários objetos
Lampada lamp1;
Lampada lamp2;
// instanciando os objetos
lamp1 = new Lampada();
lamp2 = new Lampada();
// usando os objetos
lamp1.ligar();
lamp2.desligar();
```

Identidades

Objetos que usam outros objetos

- A classe **Lampada** pode ser usada por outros objetos interessados em suas capacidades e características.
- Uma classe **LampTest** poderia ser construída para utilizar a classe **Lampada**, testando suas funcionalidades.
- Outras classes poderiam ser criadas para conter vários objetos **Lampada**, como um painel existente no mundo real.

Um Programa Simples

```
public class LampTest {
  public static void main (String args[]) {
    Lampada lamp = new Lampada();
    System.out.println(">>Lampada foi criada");
    lamp.ligar();
    System.out.println(">>Lampada foi ligada");
    lamp.desligar();
    System.out.println(">>Lampada foi desligada");
  }
}
```

POO::Dissecando uma Classe

Atributos

- Variáveis internas que pertencem a classe.
- Também conhecidos como variáveis-membro.
- Indicam valores (dados) que serão armazenados pelos objetos.
- Podem representar estados de um objeto.
- Podem ter acesso externo permitido ou não.
- São automaticamente inicializados na instanciação.

Métodos

- São operações internas implementadas para uma classe, conhecidas também como funções-membro.
- Modificam o estado de um objeto.
- Representam uma ação do objeto ou transformação ao qual este se sujeita.
- Podem receber dados externos (parâmetros) e retornar valores.
- Podem ter acesso externo permitido ou não.

Construtores

- Métodos especiais que criam objetos (criam instâncias de uma classe = objeto).
- Também preparam o objeto para uso.
- Em geral, seu uso não é restringido.
- **Tem sempre o mesmo nome que a classe.**
- Só podem ser acionados por meio do operador de instanciação **new**:

```
Lampada  lampada  =  new Lampada();
```

Construtores

- Servem para:
  - obrigar a criação de objetos com parâmetros específicos;
  - atribuir valores aos campos; e
  - executar validações iniciais (garantia de estado inicial válido).
- Toda classe possui, ao menos, um construtor:
  - se o programador não inclui um, o compilador fornece um *construtor-default* (que não faz nada e não recebe parâmetros).
  - programador pode definir vários construtores diferentes.

Construtores

```
public class Quadrado {
  private double lado;
  // construtor
  public Quadrado(double l) {
    setLado(l);
  }
  public void setLado(double l) {
    if (l<0) throw new RuntimeException();
    lado = l;
  }
  public double getLado() { return lado; }
  public double area() { return lado*lado; }
}
```

Construtores

- A instanciação sempre usa um construtor:

```
Quadrado q = new Quadrado(1.5);
Lampada l = new Lampada();
```

- Mas...
  o que acontece quando o código da classe não possui construtores?

Construtores

- Se o programador não fornece construtores, o compilador adiciona, automaticamente, um construtor default (sem parâmetros).
- Se o programador fornece qualquer construtor, o compilador não realiza qualquer intervenção na classe.
- Esta característica está presente em todas as linguagens de programação orientadas a objeto (C++, Java, C#, Scala, etc.)

Construtores (reforçando...)

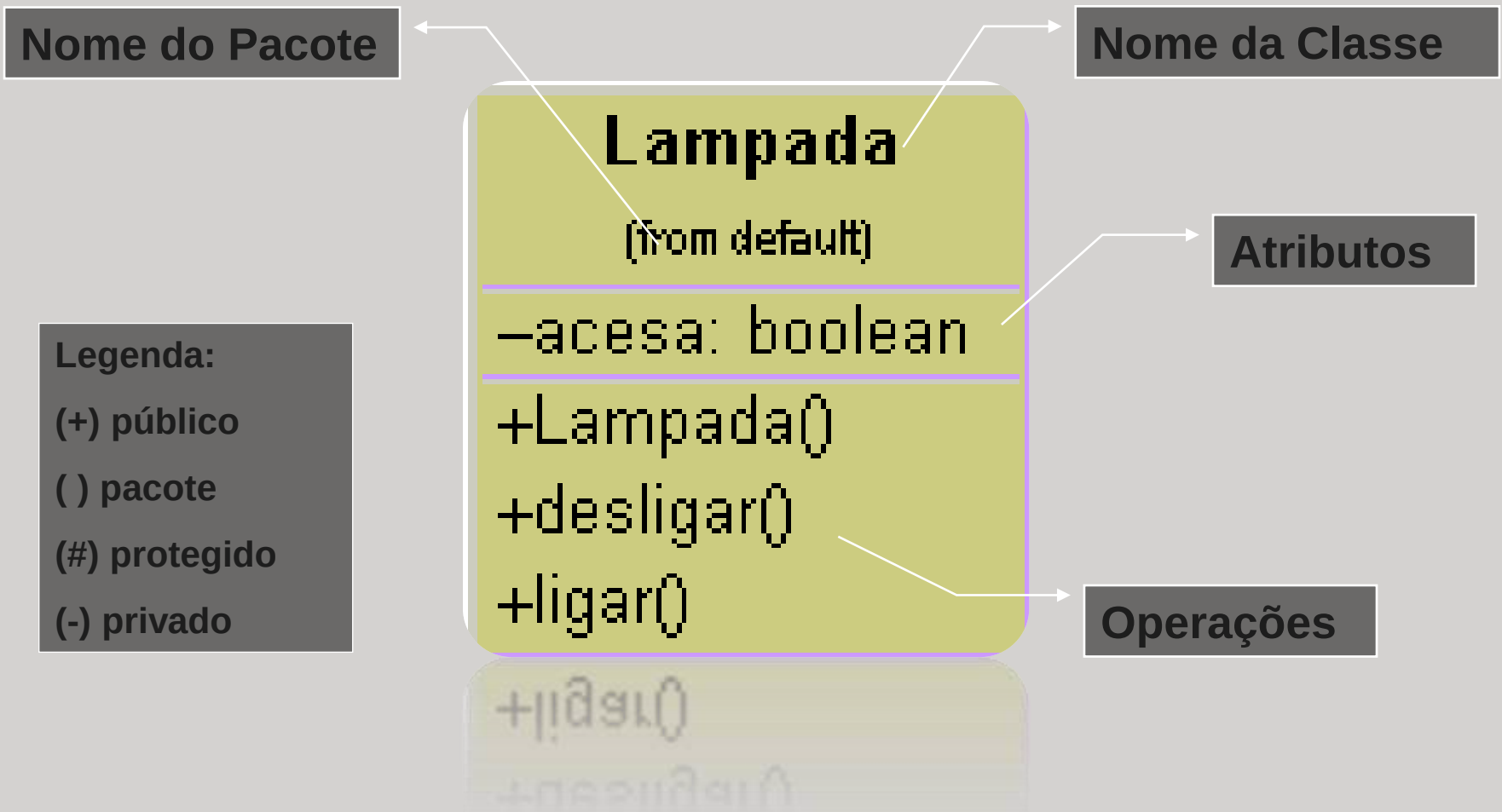
- São geralmente públicos
  (mas podem ser privados ou protegidos).
- Sempre tem o mesmo nome que a classe.
- Não tem valor de retorno.
- Se não recebem parâmetros são denominados *construtor-default*.
- Se recebem parâmetros: construtor parametrizado.

Visibilidade ou Acessibilidade

Encapsulamento

- Definida pelos *especificadores de acesso*:
  - public
  - protected
  - private
- Indicam a visibilidade do membro, ou seja, como se dá seu uso externo. Em outras palavras, constituem um mecanismo de restrição de uso.
- Permitem *encapsular* partes selecionadas da estrutura interna da classe (atributos, operações e construtores).

Notação UML (*Unified Modeling Language*)



Recomendações de Estudo

- **Acessar** o *blog*:
  - http://tecnopode.blogspot.com
- **Estudar** com livros:
  - Java – Guia do Programador, 3ª Ed, P. JANDL Jr, Novatec, 2015.
  - Java: Como Programar, 6ª Ed, DEITEL, H. & DEITEL, P., Pearson, 2007.