Universidade Federal de Uberlândia
Faculdade de Computação
Curso de Graduação de Sistemas de Informação
Introdução a Sistemas de Informação

PROGRAMAÇÃO ORIENTADA A OBJETO - POO

INTRODUÇÃO

Todos os sistemas de computação, dos mais complexos aos mais simples programas de computadores, são desenvolvidos para auxiliar na solução de problemas do mundo real. Mas, de fato, o mundo real é extremamente complexo.

INTRODUÇÃO

Quando falamos em desenvolvimento de software e seus paradigmas, existe uma sequência de aprendizado natural:

1. **Programação Imperativa**
2. **Programação Orientada a Objetos**
3. **Programação Funcional**

PROGRAMAÇÃO IMPERATIVA

É aquela que 99.99% dos desenvolvedores aprendem em seus primeiros contatos com a área de programação.

Nessa etapa, o aluno irá ver os conceitos iniciais e extremamente importantes, geralmente pautados no **"Portugol"** ou, muitas vezes, na **Linguagem C.**

Todo aplicativo/algoritmo é desenvolvido pensando-se em uma série de etapas que o software deve cumprir para atingir o seu objetivo.

Em resumo, é um **passo-a-passo (uma sequência lógica)** para se chegar a um **objetivo final.**

EXEMPLO

QUAIS SÃO AS ETAPAS PARA A TROCA DE UMA LÂMPADA?

- Desligar o interruptor;
- Pegar uma escada;
- Montar a escada;
- Subir na escada;
- Desenroscar a lâmpada queimada;
- Descer da escada;
- Jogar a lâmpada queimada no lixo;
- Pegar uma lâmpada nova;
- Subir na escada;
- Rosquear a nova lâmpada;
- Descer da escada;
- Ligar o interruptor para verificar se a nova lâmpada acende.

PORQUE APRENDER POO

O desenvolvedor novato percebe que muitas das ferramentas e linguagens atuais (Java, Ruby, Python e entre outras), são linguagens que utilizam, além do paradigma de programação imperativa, o **paradigma Orientado a Objetos.**

Hoje em dia, o mercado de programação é dominado pelo paradigma orientado a objetos. Em outras palavras, é raro você trabalhar com uma linguagem de programação atual que não suporte POO.

Sabendo POO, o desenvolvedor poderá resolver os mesmo problemas utilizando, não somente o passo-a-passo tradicional, mas também uma modelagem do problema de uma forma mais natural/real.

EXEMPLO - POO

QUAIS SÃO AS ETAPAS PARA A TROCA DE UMA LÂMPADA?

A primeira coisa a fazer é pensarmos em classes.



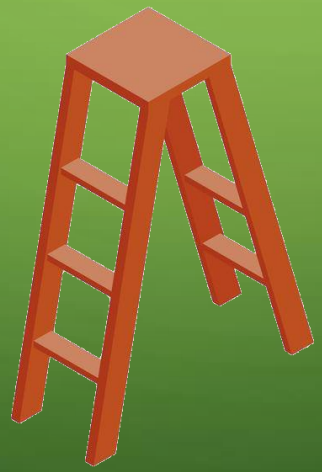
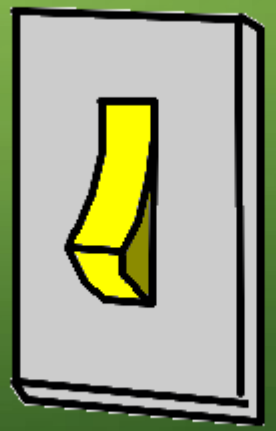
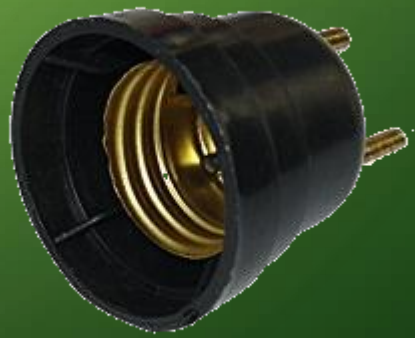
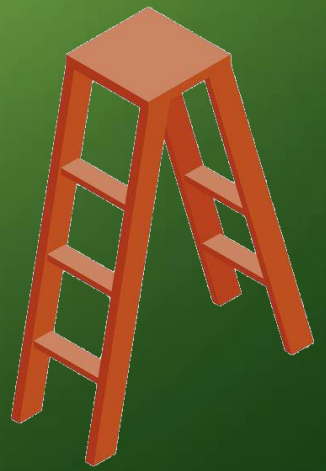
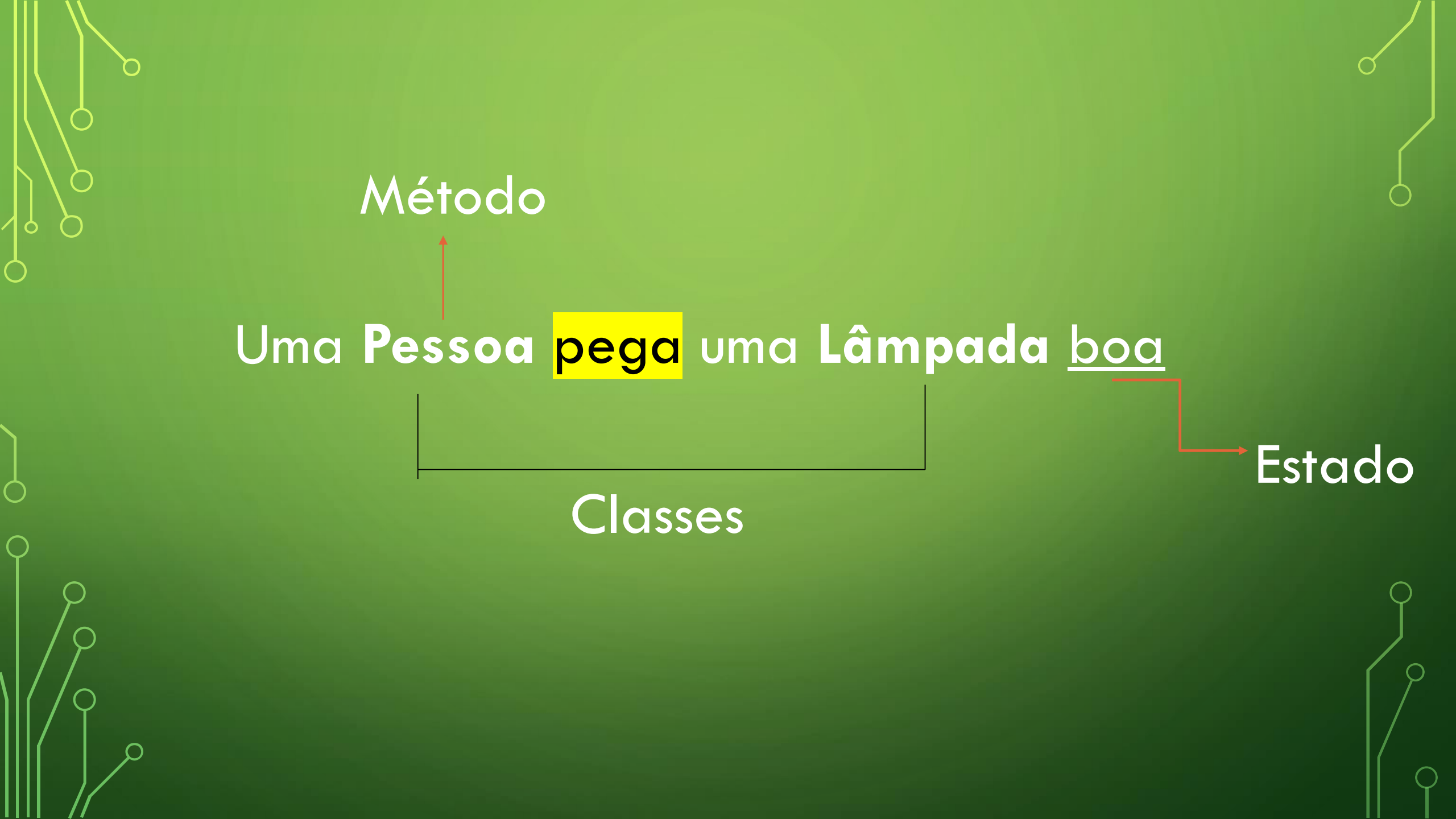




Imagem ilustrativa

CLASSE

- ATRIBUIÇÕES
- MÉTODOS (AÇÕES)
- ESTADO (ATUAL)




Método
Uma Pessoa pega uma Lâmpada boa
Estado
Classes

PRINCIPIOS BASICOS

- **Abstração:** Quando há concentração nas características essenciais, gerenciando a complexidade, ou abstraindo-a.
- **Encapsulamento:** Um mecanismo da linguagem de programação para restringir o acesso a alguns componentes dos objetos, escondendo os dados de uma classe e tornando-os disponíveis somente através de métodos.
- **Modularidade:** Decomposição de um problema em pequenos pedaços, para gerenciar complexidade
- **Herança:** Possibilidade de reuso de código o que otimiza a produção da aplicação em tempo e linhas de código.

VANTAGENS DA POO

- **Confiável:** O isolamento entre as partes gera software seguro. Ao alterar uma parte, nenhuma outra é afetada.
- **Oportuno:** Ao dividir tudo em partes, várias delas podem ser desenvolvidas em pararelo
- **Manutenível:** Atualizar um software é mais fácil. Uma pequena modificação vai beneficiar todas as partes que usarem o objeto
- **Extensível:** O software não é estático. Ele deve crescer para permanecer útil
- **Reutilizável:** Podemos usar objetos de um sistema que criamos em outro sistema futuro
- **Natural:** Mais fácil de entender. Você se preocupa mais na funcionalidade do que nos detalhes de implementação.