

# Programação Orientada a Objetos

*Introdução à POO: paradigma, abstração, objetos e classes e instanciação*

---

Ely – [ely.miranda@ifpi.edu.br](mailto:ely.miranda@ifpi.edu.br)

# Paradigma de programação

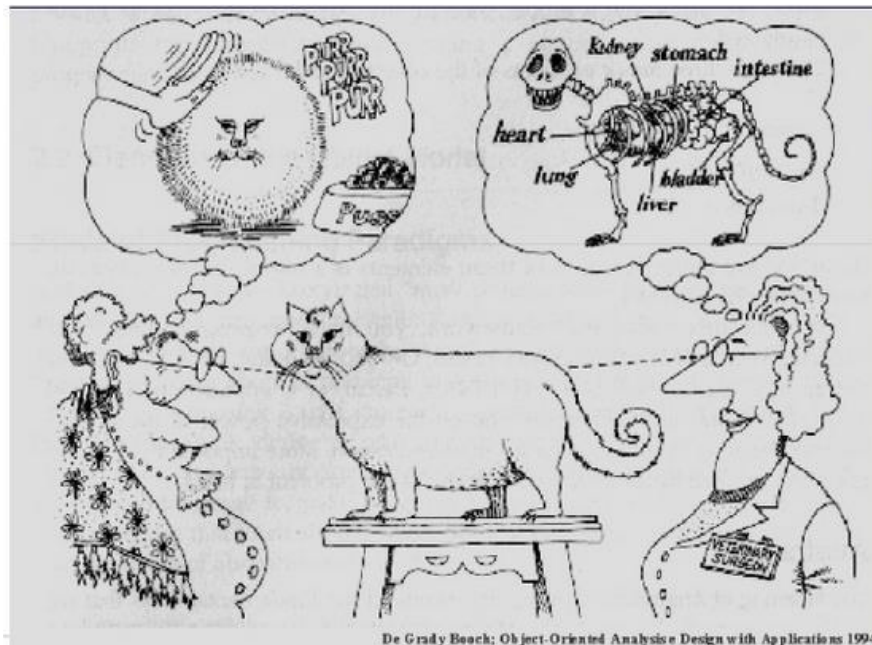
- *É um modelo, padrão ou estilo de programação suportado por linguagens que agrupam certas características comuns;*
- Foco em uma forma de pensar e resolver problemas.
- Ex:
  - Paradigma imperativo: os principais elementos são funções e módulos;
  - Paradigma orientado a objetos: os principais elementos são objetos;

# Motivação

- Por que estudar um paradigma novo:
  - Desenvolver a habilidade de aprender novas linguagens;
  - Aprofundar conceitos de algumas linguagens;
  - Compreender que linguagens são mais adequadas para alguns problemas;
  - Habilidade maior de projetar novas linguagens;
  - Abstrair melhor problemas para o ambiente computacional.

# Abstração

- Definir apenas os elementos essenciais para um sistema;
- Definir níveis de complexidade dos elementos;
- Ignorar aspectos irrelevantes relacionados à resolução do problema



# Abstração

- Dado uma instituição bancária qualquer:
  - A instituição Banco possui:
    - Estrutura física;
    - Localização;
    - Agências, funcionários...;
  - Pessoas com contas em um banco possuem:
    - Nome;
    - Altura;
    - cor dos olhos;
    - CPF...



# Abstração

- Se precisássemos criar um sistema bancário, precisaríamos abstrair detalhes;
- Abstrair é simplificar;
- Abstrair é transformar uma situação em algo computacional;
- Poucos dados dos slides anterior seriam úteis se quiséssemos controlar operações financeiras;

# Abstração

- Uma possível abstração/simplificação para uma agência bancária:
  - Um “array”/conjunto de contas cada uma com:
    - Numero: inteiro;
    - CPF do titular: número;
    - Saldo: real;
    - Operações/funções de crédito e débito;
  - Em C, tal array teria “structs” com os dados e haveria funções implementando as operações;

# Paradigmas e abstração

- Um paradigma de programação também ajuda a abstrair problemas:
  - Simplifica elementos do mundo real;
  - Transforma-os em analogias/peças aplicáveis a algoritmos e soluções:
    - Em análise: modelos visuais, fluxogramas, diagramas e etc;
    - Em programação: variáveis, ciclos, condicionais, expressões (valor, tipo), entrada e saída;





# Diversidade de linguagens

- *C, C++, Basic, COBOL, Lisp, Haskell, Modula-2, Oberon, Prolog, Java, C#, Pascal, PL/1, Ada, Smalltalk, Simula, Algol, Eiffel, Fortran, ASM, Scheme, CLOS, Maude, Python, Glass, Ruby, JavaScript, Perl, Lua, Groovy, SQL, Earlang e etc*
- Por que tantas?
  - Propósitos Diferentes;
  - Avanços Tecnológicos;
  - Interesses comerciais;
  - Cultura e background científico.

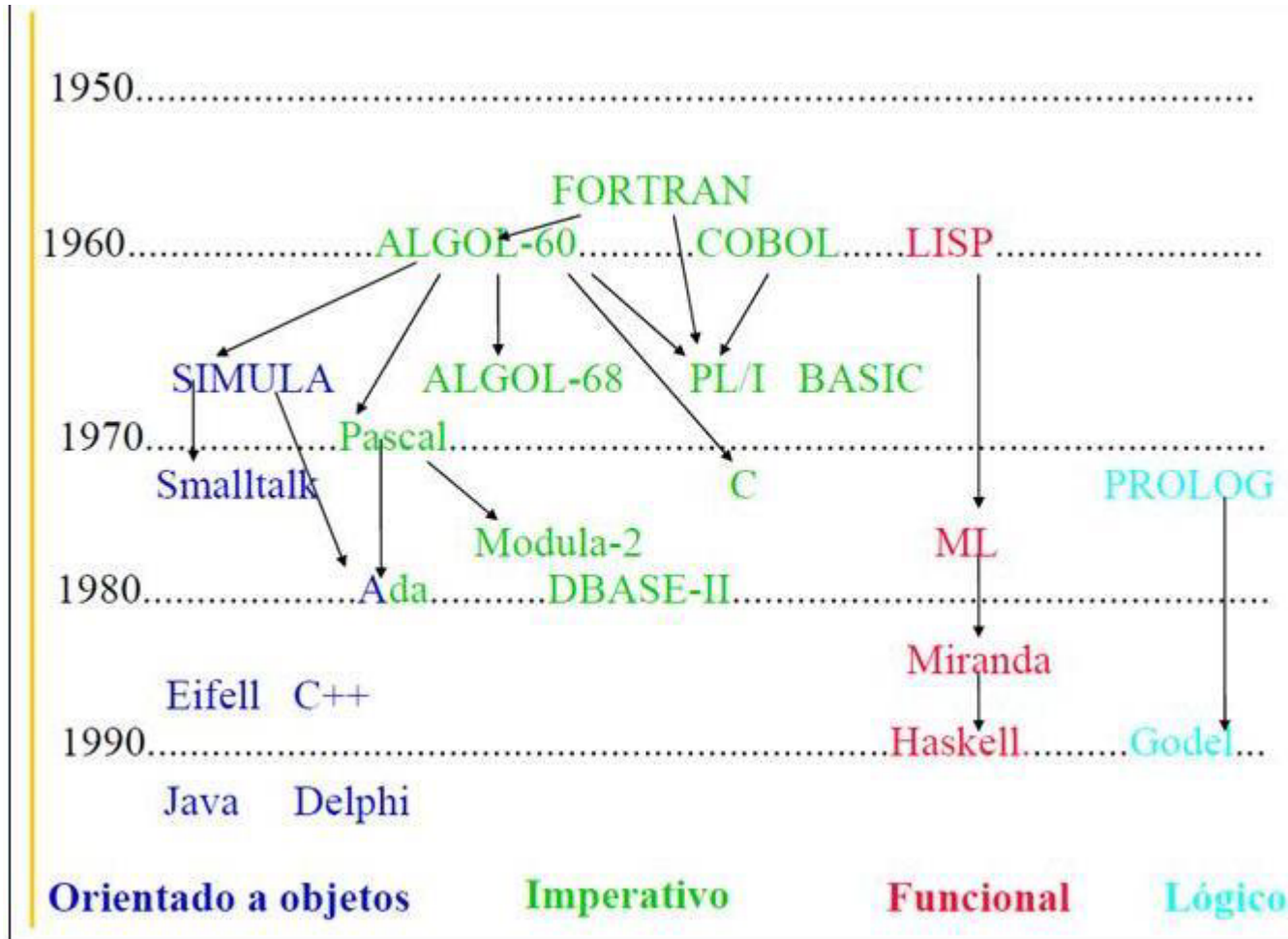
# Alguns Paradigmas

- Imperativo/procedural: C, Pascal, COBOL...
- Orientado a Objetos: C++, Java, Ruby...
- Funcional: Haskell, F#
- Lógico: Prolog, LISP...


*Tendência: linguagens com recursos de mais de um paradigma;*

# Alguns Paradigmas

- Uma linha do tempo resumida:



# Orientação a Objetos

- Paradigma que envolve várias etapas de desenvolvimento de software, dentre elas:
  - Projeto;
  - Análise;
  - Programação;  Foco deste curso
- Possui relevante aceitação comercial e acadêmica:

Category	Ratings Oct 2013	Delta Oct 2012
Object-Oriented Languages	56.1%	-1.5%
Procedural Languages	36.7%	-1.2%
Functional Languages	3.6%	+0.4%
Logical Languages	3.6%	+2.3%

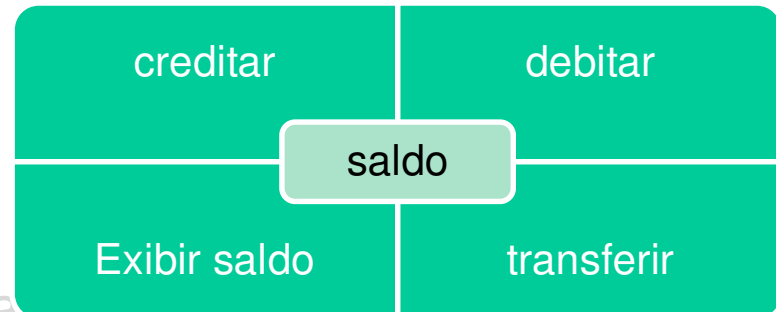
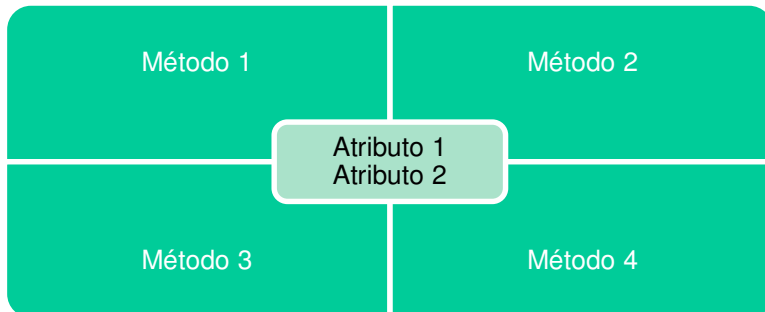
Fonte: Ranking Tiobe (<http://www.tiobe.com/>)

# Orientação a Objetos

- Principais elementos:
  - Classes e Objetos;
  - Herança;
  - Encapsulamento;
  - Polimorfismo.

# Orientação a Objetos

- Parte do princípio que o mundo é formado por elementos (objetos) que interagem entre si;
- Tudo relacionado ao paradigma são objetos:
  - Possuem características/estado: variáveis conhecidas como atributos;
  - Possuem comportamentos: funções denominadas métodos;
  - Os métodos alteram os valores dos atributos;



# Objetos

- São analogias a elementos do mundo real:
  - Pessoa, Motor, Lâmpada, Cartão de crédito, Processo, Conta, Produto, Venda...
- Possuem **características** e **comportamentos**:
  - Pessoa: **cor**, **altura**, **CPF**, **contas**, **comprar**, **vender**, **postar**, **comentar**;
  - Blog: **postagens**, **nome**, **URL**, **exibir postagens**, **excluir comentários**, **pesquisar**;
  - Conta bancária: **saldo**, **titular**, **creditar**, **debitar**, **transferir**;
  - Jogador: **número**, **nome**, **velocidade**, **nível de cansaço**, **chutar**, **correr**, **lançar**, **efetuar passe**.

# Definindo Objetos

- Objetos: Substantivos
  - Atributos:
    - Características relacionadas aos substantivos;
  - Métodos:
    - Verbos que representam comportamentos e ações dos objetos;



# Atributos

- Características e propriedades que os objetos possuem;
- Exemplos:
  - Pessoa: cor, altura, CPF, contas;
  - Blog: postagens, nome, URL;
  - Conta bancária: número, saldo, titular;
  - Jogador: número, nome, velocidade, nível de cansaço;
- Preferencialmente devem ser alterados e lidos apenas por métodos.

# Atributos

- Definem ainda o estado dos objetos;
- São definidos em linguagens de programação como tipos inteiros, arrays, reais, strings...

# Métodos

- Comportamentos de um objeto ou ações que um objeto pode realizar;
- Exemplos:
  - Pessoa: comprar, vender, postar, comentar;
  - Blog: exibir postagens, excluir comentários, pesquisar;
  - Conta bancária: creditar, debitar, transferir;
  - Jogador: chutar, correr, lançar, efetuar passe;
- São implementados através conjuntos de instruções (semelhante à funções);

# Métodos

- Métodos podem executar ações sobre outros objetos;
- Devem ser os responsáveis pela alteração dos atributos;

# Vantagens

- Objetos são elementos padronizados e podem ser reutilizados;
  - Analogia com o mundo industrial:
    - Produção em larga escala de peças;
    - Peças padrões são mais baratas;
    - São mais testadas e conseqüentemente mais confiáveis;
    - Possuem facilidades de manutenção e substituição;

# Vantagens

- Modularidade: cada objeto é um módulo;
- Bibliotecas e frameworks: Agrupamento de objetos relacionados resolvendo problemas maiores;
- Reutilização: objetos usados em diferentes contextos/sistemas;
- Extensão: objetos simples podem ser estendidos através de herança;
- Confiabilidade: possibilidade de criar testes simples e complexos;

*Resumindo: menores custos de desenvolvimento;*

# Desvantagens

- Novos conceitos;
- Maior curva de aprendizado;
- Programadores “antigos” possuem vícios de outras linguagens;
- Aumento da complexidade de software:
  - Necessidade de ferramentas que deem mais produtividade;
  - Ambientes de desenvolvimento não acompanham tão rápido as “novidades”;

# Classes

- Enquadramento dos objetos em categorias conforme atributos e métodos;
- Diz-se que uma classe é uma “matriz/origem” de objetos;
- São modelos a partir dos quais os objetos são criados (instanciados);
- Analogias:
  - Planta (classe) e casa construída (objeto);
  - Projeto (classe) e execução (objeto);
  - Receita (classe) e bolo (objeto).



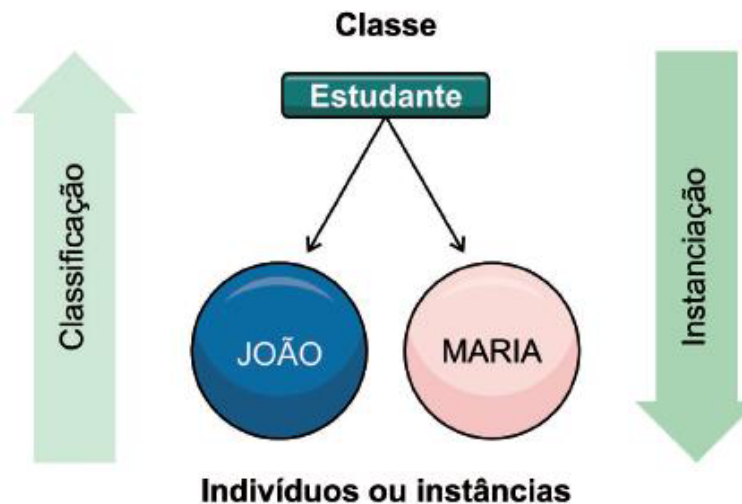
# Classes x Objetos

- Classes são modelos, objetos são classes em execução/memória (instanciadas);
- Uma classe está para um objeto, assim como:
  - Uma receita está para uma torta;
  - Uma planta está para uma casa;

Classe Pessoa	Objeto Pessoa
Nome: Texto; Data de Nascimento: Data; Altura: Número;	Nome: Cláudio; Data de Nascimento: 20/05/1978; Altura: 1.6

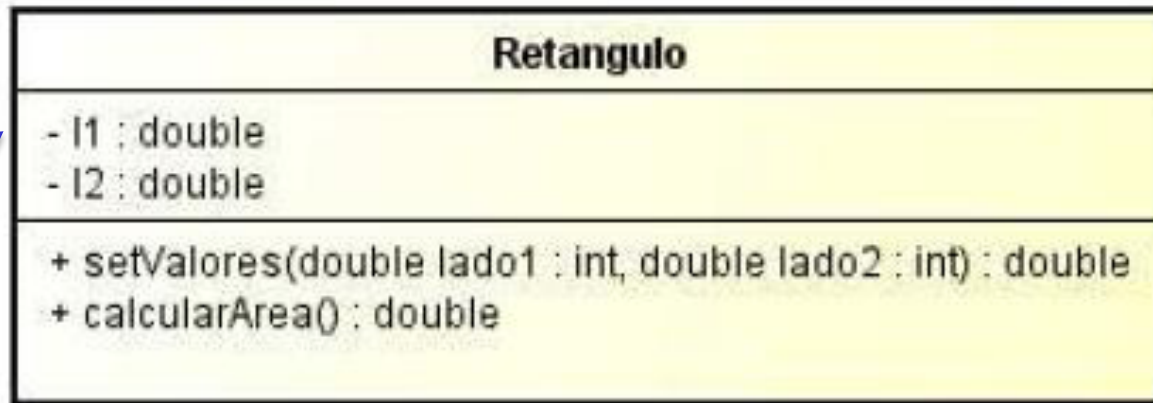
# Classificação x Instanciação

- Classificar:
  - Definir classes, ou seja, agrupar objetos com atributos e métodos semelhantes;
- Instanciar:
  - Criar objetos a partir de classes



# Classes em UML

- Unified Modeling Language



atributos

métodos

# Exemplo de Classe em C++

```
class Retangulo {  
    int l1, l2;  
public:  
    void set_valores (int,int);  
    int calcula_area () {  
        return l1 * l2;  
    }  
};  
  
void Retangulo::set_valores (int lado1, int lado2) {  
    l1 = lado1;  
    l2 = lado2;  
}
```

Atributos (estado)

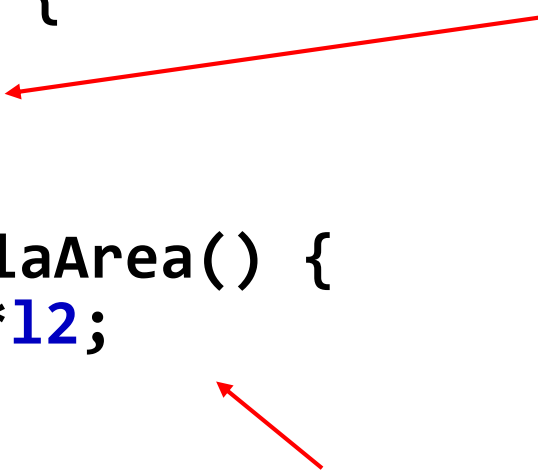
Métodos (comportamento)

# Exemplo de Classe em Java

```
class Retangulo {  
    double l1;  
    double l2;  
  
    double calculaArea() {  
        return l1*l2;  
    }  
}
```

Atributos (estado)

Método (comportamento)



# Instanciação de objetos em Java

- Para criarmos um objeto, devemos realizar uma instanciação;
- Instanciar um objeto é o equivalente a:
  - alocar uma área de memória;
  - atribuímos a uma variável o endereço dessa área.
- Dizemos que um variável é uma **referência para um objeto**;
- Instanciamos um objeto a partir do nome de sua classe e usando o operador **new**;

# Instanciação de objetos em Java

- Sintaxe:

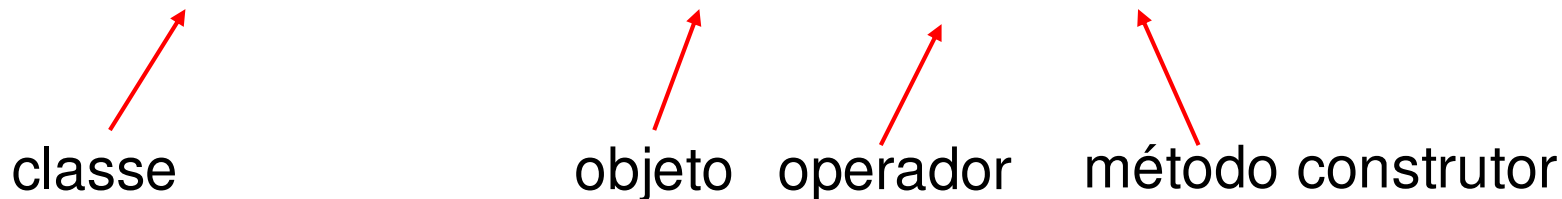
- `<NomeDaClasse> <nomeDoObjeto>;`
- `<nomeDoObjeto> = new <NomeDaClasse()>;`

- Ex1:

```
Retangulo retangulo;  
retangulo = new Retangulo();
```

- Ex2:

```
Retangulo retangulo = new Retangulo();
```

  
classe                      objeto    operador    método construtor

# Exemplo de Classe em Java

- Classe que utiliza a classe Retangulo:

```
public class TestaRetangulo {  
    public static void main(String args) {  
        Retangulo retangulo = new Retangulo();  
  
        retangulo.l1 = 10;  
        retangulo.l2 = 20;  
  
        System.out.println(retangulo.calculaPerimetro());  
    }  
}
```

*Atributos e métodos de uma classe são acessados através de um ponto “.”*



# Compilando e executando as classes

- Coloque as classes Retangulo e TestaRetangulo em um mesmo diretório;
- Execute o comando para compilá-las:  
`javac *.java`
- Execute o arquivo TestaRetangulo.class:  
`java TestaRetangulo`

*Um detalhe: a classe que possui o método main é instanciada automaticamente pelo ambiente java*

# Programação Orientada a Objetos

*Introdução à POO: paradigma, abstração, objetos e classes e instanciação*

---

Ely – [ely.miranda@ifpi.edu.br](mailto:ely.miranda@ifpi.edu.br)