



Programação Orientada a Objetos

Aula 05 – Conceitos da Orientação a Objetos

Prof. Luiz Mário Lustosa Pascoal



Objetivos da aula

- Projetos em SI
- Histórico do Paradigma OO
- Princípios de OO

- 
- *“Coisas simples devem ser simples e coisas complexas devem ser possíveis.”*

Alan Kay



Sistemas de Informações

- **A necessidade é a mãe das invenções**

- Em consequência do crescimento da importância da informação, surgiu a necessidade de gerenciar informações de uma forma adequada e eficiente e, desta necessidade, surgiram os denominados ***sistemas de informações***.

- Um SI é uma combinação de pessoas, dados, processos, interfaces, redes de comunicação e tecnologia que interagem com o objetivo de dar suporte e melhorar o processo de negócio de uma organização com relação às informações.

- Vantagens do ponto de vista competitivo.

- Objetivo principal e final da construção de um SI: ***adição de valor à organização***.



Sistemas de Software

- Um dos componentes de um sistema é denominado **sistema de software**.

- Compreende os módulos funcionais computadorizados que interagem entre si para proporcionar a automatização de diversas tarefas.

- Característica intrínseca do desenvolvimento de sistemas de software: **complexidade**.

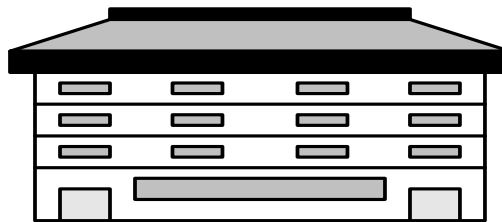
- Como minimizá-la?

Sistemas de Software

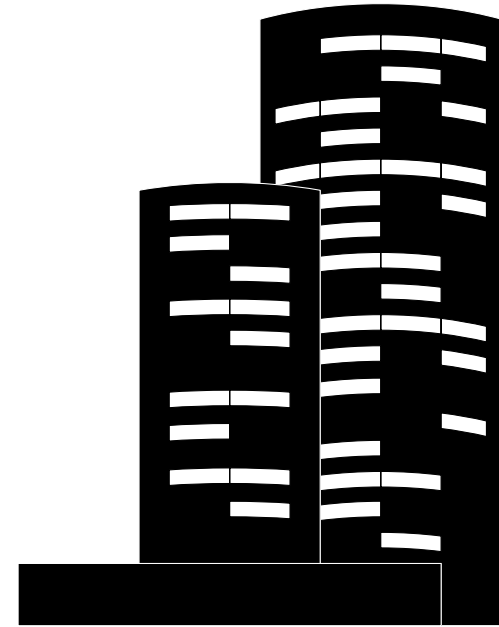
Uma analogia...



Casa de canhorro

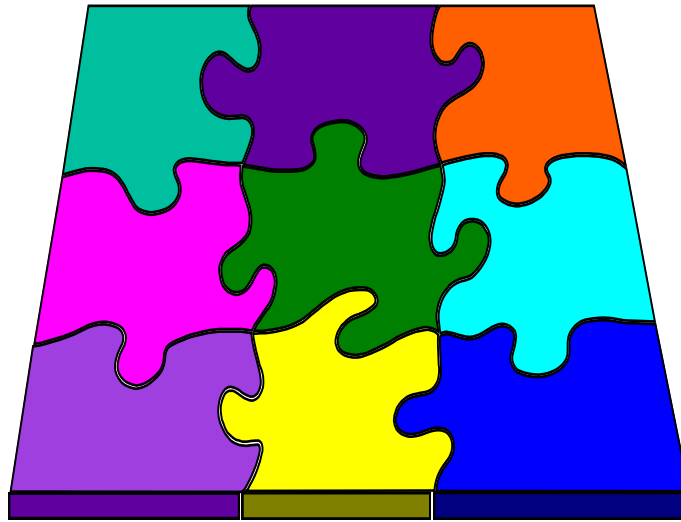


Casa



Arranha-Ceús

Aumento da complexidade



O paradigma de orientação a objetos



Paradigma?

- *Um paradigma é uma forma de abordar um problema.*
- No contexto da modelagem de um sistema de software, um paradigma tem a ver com a forma pela qual esse sistema é **entendido e construído**.
- A primeira abordagem usada para modelagem de sistemas de software foi o ***paradigma estruturado***.
 - Uso da técnica de *decomposição funcional*
 - “divida sucessivamente um problema complexo em subproblemas”
- Hoje em dia, praticamente suplantou o paradigma anterior, o ***paradigma da orientação a objetos***...



O Paradigma de Orientação a Objetos

● A **orientação a objetos** é um paradigma de análise, projeto e programação de sistemas de *software* baseado na composição e interação entre diversas unidades de software chamadas de objetos.”

- Smaltalk, foi criada por Alan Kay
- Simula67, criada por Ole Johan Dahl em 1967
- Java, o Visual Basic, o Object Pascal (Delphi)

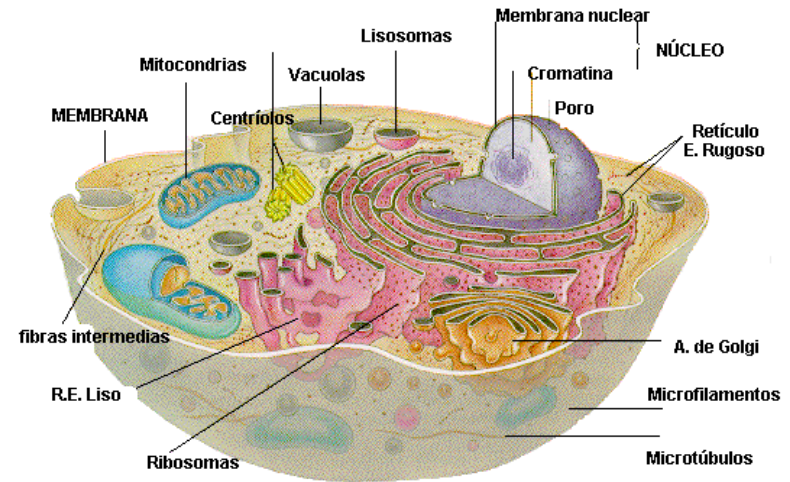
O Paradigma de Orientação a Objetos

- O paradigma da OO surgiu no fim dos anos 60.
- Alan Kay, um dos pais desse paradigma, formulou a chamada **analogia biológica**.
- *“Como seria um sistema de software que funcionasse como um ser vivo?”*



Analogia Biológica


- Cada “célula” interagiria com outras células através do envio de mensagens para realizar um objetivo comum.
- Adicionalmente, cada célula se comportaria como uma unidade autônoma.
- De uma forma mais geral, Kay pensou em como construir um sistema de software a partir de agentes autônomos que interagem entre si.





Fundamentos da Orientação a Objetos

- Através de sua analogia biológica, Alan Kay definiu os fundamentos da orientação a objetos.
 1. Qualquer coisa é um objeto.
 2. Objetos realizam tarefas através da requisição de serviços a outros objetos.
 3. Cada objeto pertence a uma determinada **classe**. Uma classe agrupa objetos similares.
 4. A classe é um repositório para comportamento associado ao objeto.
 5. Classes são organizadas em hierarquias.



O paradigma de orientação a objetos visualiza um sistema de software como uma coleção de agentes interconectados chamados ***objetos***. Cada objeto é responsável por realizar tarefas específicas. É através da interação entre objetos que uma tarefa computacional é realizada.

Um sistema de software orientado a objetos consiste de objetos em colaboração; com o objetivo de realizar as funcionalidades deste sistema. Cada objeto é responsável por tarefas específicas. É através da cooperação entre objetos que a computação do sistema se desenvolve.



Programação Orientada a Objetos

- Histórico
- Smalltalk (everything is a object)
 - Linguagem OO desenvolvida na Xerox.
 - Programas são compilados para bytecodes e executados em uma máquina virtual.
 - Criação do conceito de Garbage Collector
 - No Smalltalk-80 o código de bytecodes era compilado para a arquitetura nativa.
 - Ideias que Java e .NET hoje implementam. Conceito de dynamictranslation.



Conceitos e Princípios da OO

- Conceitos

- Classe
- Objeto
- Mensagem

- Princípios

- Encapsulamento
- Polimorfismo
- Generalização (Herança)
- Composição



Classes, objetos e mensagens

- O mundo real é formado de coisas.
- Na terminologia de orientação a objetos, estas coisas do mundo real são denominadas *objetos*.
- Seres humanos costumam agrupar os objetos para entendê-los.
- A descrição de um grupo de objetos é denominada ***classe de objetos***, ou simplesmente de ***classe***.

O que é uma classe?

- Uma **classe** é um molde para objetos. Diz-se que um objeto é uma instância de uma classe.

- Uma classe é uma **abstração** das características **relevantes** de um grupo de coisas do mundo real.

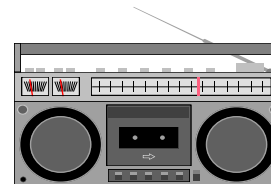
- Na maioria das vezes, um grupo de objetos do mundo real é **muito complexo** para que **todas** as suas características e comportamento sejam representados em uma classe.



Representante



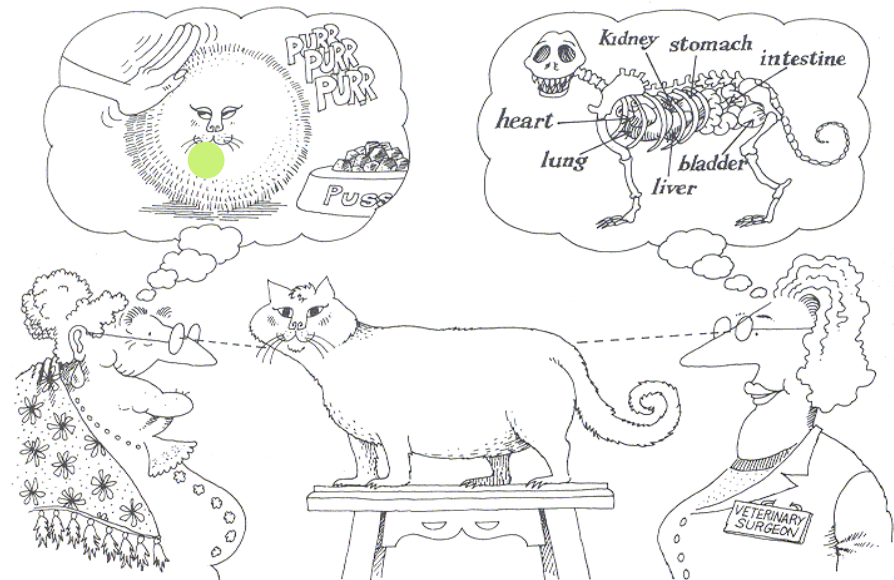
Cliente



Produto

Abstração

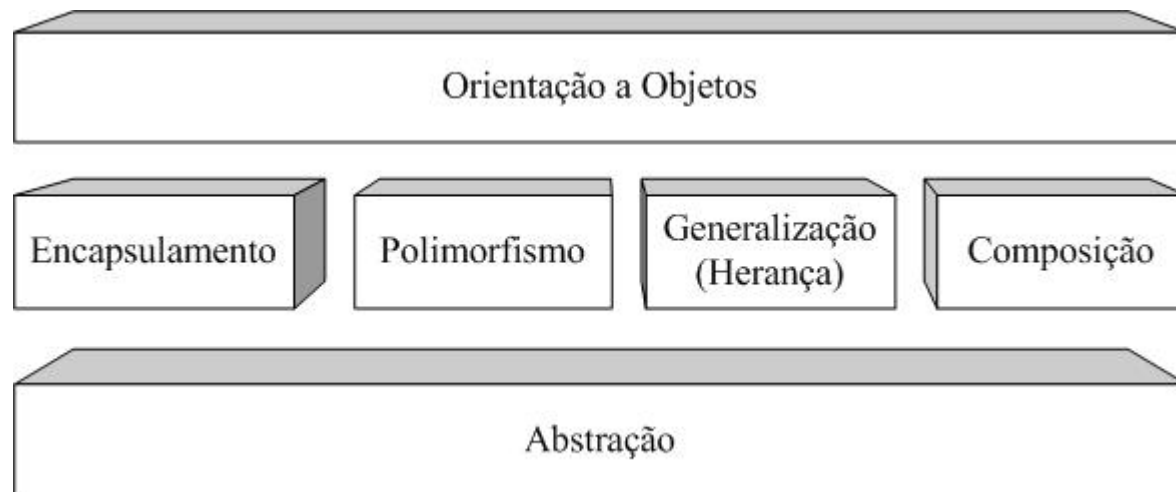
- Uma **abstração** é qualquer modelo que inclui os aspectos **relevantes de alguma coisa**, ao mesmo tempo em que ignora os menos importantes. *Abstração depende do observador.*



Abstraction focuses upon the essential characteristics of some object, relative to the perspective of the viewer.

Abstração na orientação a objetos

- A orientação a objetos faz uso intenso de abstrações.
 - Os princípios da OO podem ser vistos como aplicações da abstração.
- **Princípios da OO:** encapsulamento, polimorfismo, herança e composição,





Objetos como abstrações

- Uma abstração é uma representação das características e do comportamento relevantes de um conceito do mundo real para um determinado problema.
- Dependendo do contexto, um mesmo conceito do mundo real pode ser representado por diferentes abstrações.
 - Carro (para uma transportadora de cargas)
 - Carro (para uma fábrica de automóveis)
 - Carro (para um colecionador)
 - Carro (para uma empresa de kart)
 - Carro (para um mecânico)




Objetos como abstrações

Ex: Pessoa

- Nome
- DataNascimento
- Altura
- Peso

Classe X Objeto

- Objetos são abstrações de entidades que existem no mundo real.
- Classes são definições estáticas, que possibilitam o entendimento de um grupo de objetos. 
- **CUIDADO:** estes dois termos muitas vezes são usados indistintamente em textos sobre orientação a objetos.



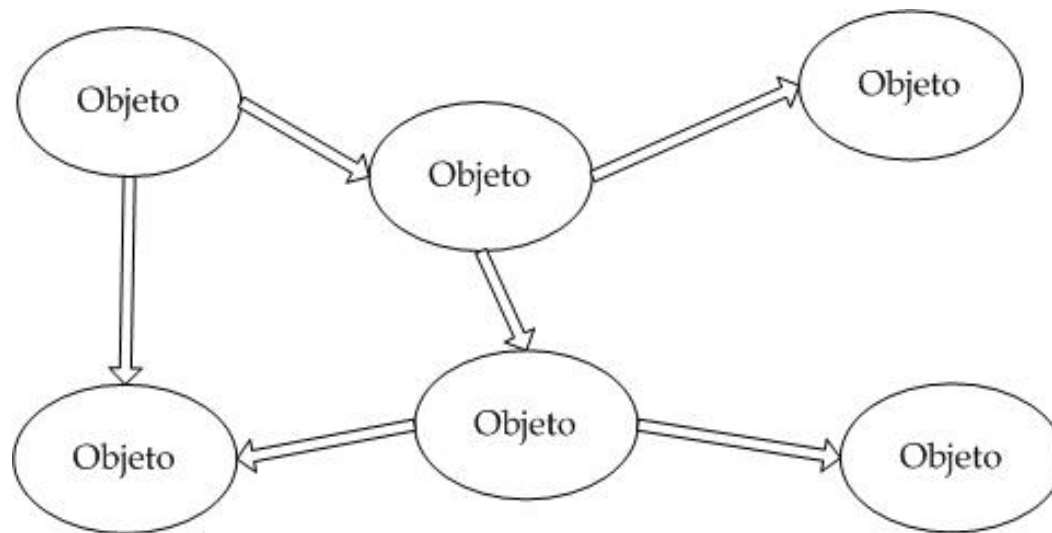
Mensagens

- Para que um objeto realize alguma tarefa, deve haver um estímulo enviado a este objeto.
- Pense em um objeto como uma entidade ativa que representa uma abstração de algo do mundo real
 - Então faz sentido dizer que tal objeto pode responder a estímulos a ele enviados
 - Assim como faz sentido dizer que seres vivos reagem a estímulos que eles recebem.
- Independentemente da origem do estímulo, quando ele ocorre, diz-se que o objeto em questão está recebendo uma ***mensagem***.
- Uma **mensagem** é uma requisição enviada de um **objeto a outro** para que este último realize alguma **operação**.

Mensagens

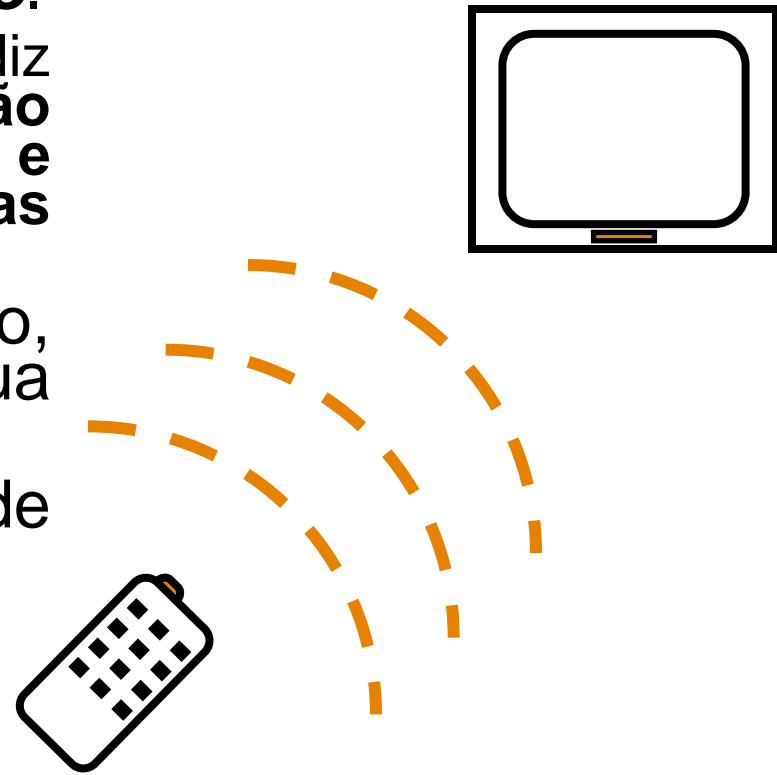
Objetos de um sistema trocam mensagens

- isto significa que estes objetos estão enviando mensagens uns aos outros com o objetivo de realizar alguma tarefa dentro do sistema no qual eles estão inseridos.



Encapsulamento

- Objetos possuem ***comportamento***.
 - O termo **comportamento** diz respeito a **que operações são realizadas por um objeto e também de que modo estas operações são executadas.**
- De acordo com o encapsulamento, objetos devem “esconder” a sua complexidade...
- Esse princípio aumenta qualidade do SOO, em termos de:
 - Legibilidade
 - Clareza
 - Reuso



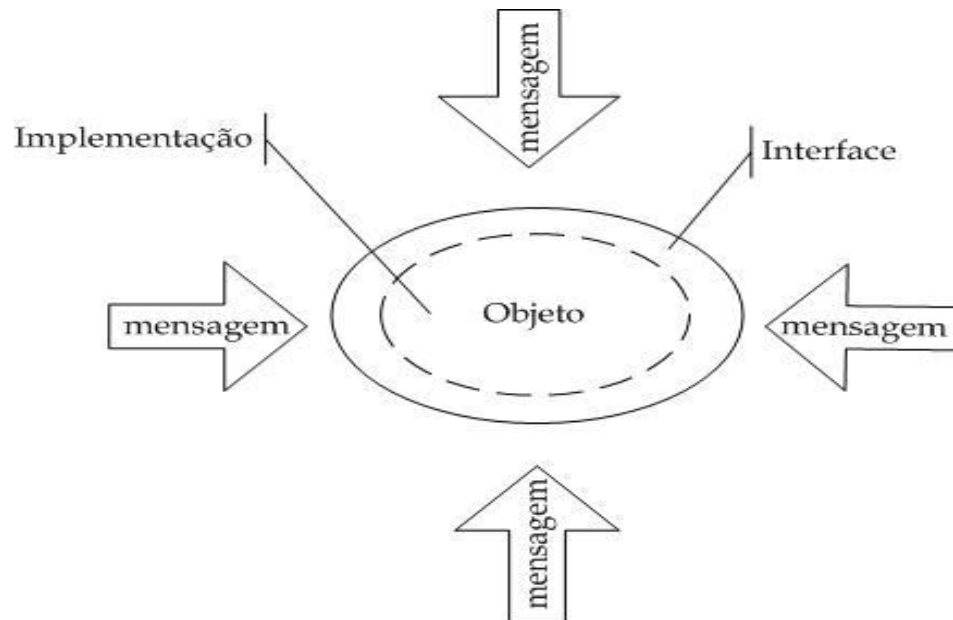


Encapsulamento

- O *encapsulamento* é uma forma de **restringir o acesso** ao comportamento interno de um objeto.
 - Um objeto que precise da colaboração de outro para realizar alguma tarefa **simplesmente** envia uma mensagem a este último.
 - O método (maneira de fazer) que o objeto requisitado usa para realizar a tarefa não é conhecido dos objetos requisitantes.
- Na terminologia da orientação a objetos, diz-se que um objeto possui uma ***interface***.
 - **A interface de um objeto é o que ele conhece e o que ele sabe fazer**, sem descrever *como* o objeto conhece ou faz.
 - A interface de um objeto define os serviços que ele pode realizar e conseqüentemente as mensagens que ele recebe.

Encapsulamento

- Uma interface pode ter várias formas de ***implementação***.
- Mas, pelo princípio do encapsulamento, a implementação utilizada por um objeto receptor de uma mensagem não importa para um objeto remetente da mesma.

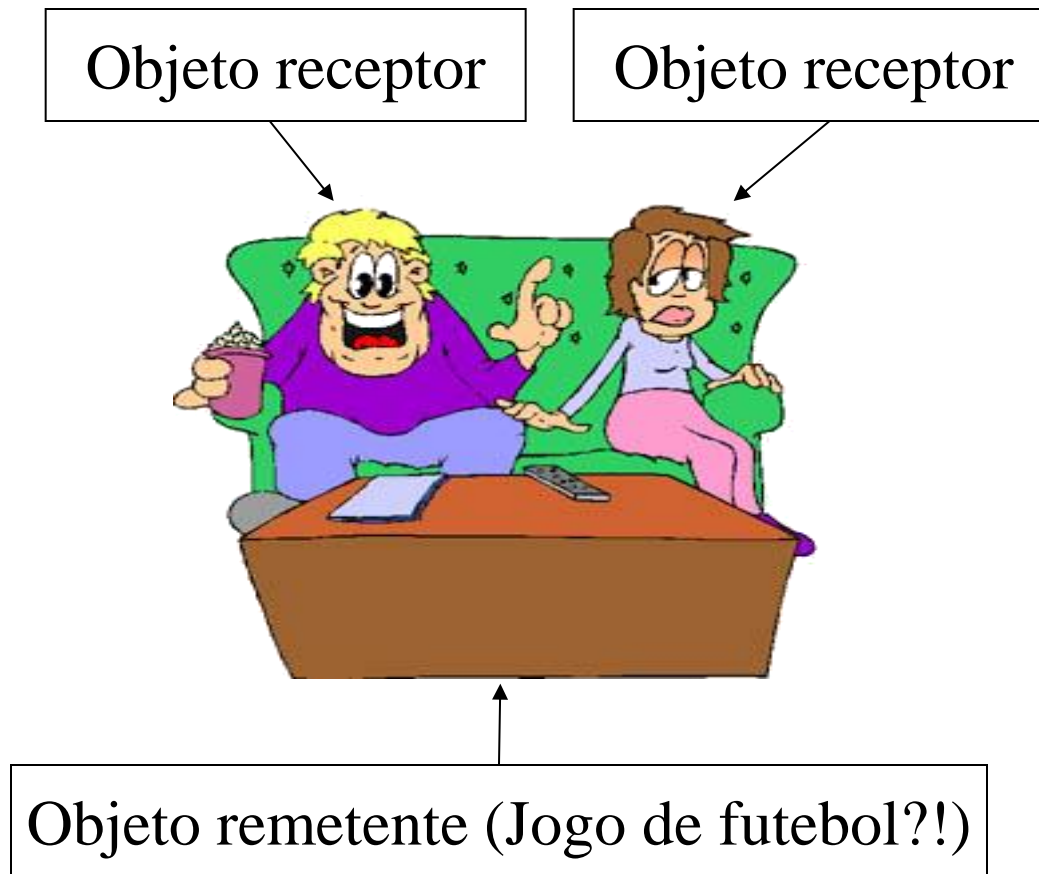




Encapsulamento

- Exemplo: Implementação de uma calculadora
 - Pessoa1: Como funciona a função de somar?
 - Pessoa2: Você passa dois parâmetros e eu te retorno o resultado
 - Pessoa1: Mas como é implementado internamente?
 - Pessoa2: Você não precisa saber disso

Polimorfismo





Polimorfismo

É a habilidade de objetos de classes diferentes responderem a mesma mensagem de diferentes maneiras.

Em uma linguagem orientada a objetos:

```
for(i = 0; i < poligonos.tamanho(); i++)  
    poligonos[i].desenhar();
```



Polimorfismo

- Um exemplo: Capacidade de objetos derivados de uma mesma base reagirem de forma diferente
- Exemplo
 - Classe Animal possui operação andar
 - Classe Pessoa derivada de animal, quando andar, vai andar da sua forma
 - Classe Cachorro derivada de animal, quando andar, vai andar da sua forma



Generalização (Herança)

- A herança pode ser vista como um nível de abstração acima da encontrada entre classes e objetos.
- Na herança, classes semelhantes são agrupadas em hierarquias.
 - Cada nível de uma hierarquia pode ser visto como um nível de abstração.
 - Cada classe em um nível da hierarquia herda as características das classes nos níveis acima.

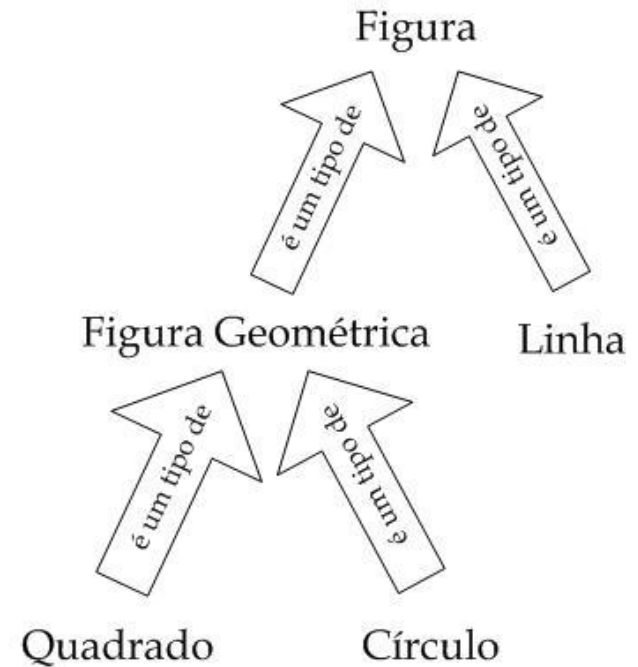
Herança

- A herança facilita o compartilhamento de comportamento entre classes semelhantes.
- As diferenças ou variações de uma classe em particular podem ser organizadas de forma mais clara.

Maior
abstração



Menor
abstração





Coesão e Acomplamento

- Coesão e Acoplamento: conceitos importantes
- Exemplo: Construir classe Calculadora com operação de somar, que deve retornar o resultado da soma



Coesão e Acomplamento

- Coesão

- **Código não coeso** é: implementação do módulo Calculadora e da operação somar. Além de somar envia e-mail para o usuário com o resultado da operação e mostra mensagem na tela com o resultado.

- Problema: manda e-mail e mostra mensagem em tela, desnecessário para o requisitado



Coesão

- Coesão
- Perguntar sempre: Cada entidade e cada funcionalidade realizam o seu trabalho? Ou fazem mais do que devem?
- Buscar: código mais coeso possível (mais simples possível)
- Código sendo coeso, facilita a reutilização



Acomplamento

- **Código não acoplado é:** programador implementa o módulo calculadora no mesmo arquivo da tela de entrada de dados
- Problema: a calculadora pode ser reutilizada em outro projeto, mas acaba levando uma tela de entrada de dados desnecessária.
- Buscar: código menos acoplado possível
- Código menos acoplado facilita a reutilização