

INSTITUTO FEDERAL
GOIÁS

INSTITUTO FEDERAL DE GOIÁS
CÂMPUS GOIÂNIA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO
PROGRAMAÇÃO ORIENTADA A OBJETO I

Nome do Aluno: João Manoel Leite Ribeiro Nogueira Data: 18/09/2016

Prof. Renan Rodrigues de Oliveira

Trabalho: Introdução a POO

RESUMO

Este trabalho tem por objetivo introduzir os conceitos básicos de POO (Programação Orientada a Objetos). Para tanto, inicialmente demonstraremos as diferenças básicas entre programação estruturada e programação orientada a objetos. Depois, definiremos o básico da programação orientada a objetos e traremos exemplos de modelagem a objetos: um automóvel e uma lâmpada a venda num supermercado. Por fim, concluiremos sobre programação orientada a objetos demonstrando o quanto ela é útil no desenvolvimento de programas.

INTRODUÇÃO

As diferenças básicas entre programação orientada a objetos e programação estruturada:

Programação Orientada a Objetos	Programação Estruturada
Visualiza e representa o mundo real como um conjunto de objetos que interagem entre si para que determinadas operações sejam realizadas. Logo, a programação orientada a objetos define uma programação voltada aos conceitos de classes e herança.	A programação é entendida como uma sequência de passos onde um problema é resolvido a partir de um início e fim bem definidos e eventualmente dividido em sub-rotinas. Logo, a programação estruturada é voltada a procedimentos e funções definidas pelo usuário.
Enfoque em Objetos: <ul style="list-style-type: none">➤ Objetos do mundo real transforma-se em objetos no software;➤ Objetos existentes no mundo real podem se complexos, tornando necessário abstrair as características relevantes de cada entidade para o sistema em desenvolvimento;➤ O processo de abstração é fundamental para o desenvolvimento de softwares orientado a objetos.	Enfoque a Programas: <ul style="list-style-type: none">➤ Visão tradicional usa a perspectiva de algoritmo;➤ O principal bloco de construção é o procedimento ou função;➤ Conduz o foco de atenção para questões referentes ao controle e a decomposição de algoritmos maiores em outros menores;➤ Modelagem de dados divide as informações em tabelas, criando mecanismos para junção posterior.

Programação Orientada a Objetos	Programação Estruturada
<p>Programação Orientada a Objetos</p> <p>The diagram illustrates POO with three separate objects, each labeled 'Dados Objeto'. Each object is connected to a set of methods labeled 'Método'. Ellipses (...) indicate that there can be more than one object and more than one method per object.</p>	<p>Programação Estruturada</p> <p>The diagram illustrates structured programming with a single large block labeled 'Dados Globais' (Global Data). This block is connected to a vertical list of procedures labeled 'Procedimento'. Ellipses (...) indicate that there can be multiple procedures.</p>
Métodos	Procedimentos e funções
Instâncias de variáveis	Variáveis
Mensagens	Chamadas a procedimentos e funções
Classes	Tipos de dados definidos pelo usuário
Herança	Não disponível
Polimorfismo	Não disponível

FUNDAMENTAÇÃO TEÓRICA / REVISÃO DA LITERATURA

Conceitos básicos da POO (Programação Orientada a Objetos):

➔ ABSTRAÇÃO:

Construção de um modelo para representação de uma realidade, com foco nos aspectos essenciais. Logo, este princípio é uma forma de abstrair o quão complexo é um método ou rotina de um sistema, ou seja, o usuário não necessita saber todos os detalhes de como sua implementação foi realizada, apenas para que serve determinada rotina e qual o resultado esperado da mesma. Sendo assim, podemos também dividir internamente problemas complexos em problemas menores, onde resolvemos cada um deles até encontrarmos a solução do problema inteiro. Um exemplo da vida real para ilustrar esse conceito seria o conceito de carro a abstração de um veículo, que é utilizado como meio de transporte por várias pessoas para mover-se de um ponto a outro. Não é necessário que a pessoa informe que irá se locomover com a ajuda de um veículo movido a


combustível, contendo rodas e motor. Basta a pessoa informar que utilizará um carro para tal, pois esse objeto é conhecido por todos e abstrai toda essa informação por trás disso.

Na POO, uma classe é uma abstração de entidades existentes no domínio do sistema de software:

Realidade	Modelo			
	<table><tr><th>Bicicleta</th></tr><tr><td><ul style="list-style-type: none">- marcha : int- velocidade : float</td></tr><tr><td><ul style="list-style-type: none">+ mudarMarcha() : void+ mudarVelocidade() : void</td></tr></table>	Bicicleta	<ul style="list-style-type: none">- marcha : int- velocidade : float	<ul style="list-style-type: none">+ mudarMarcha() : void+ mudarVelocidade() : void
Bicicleta				
<ul style="list-style-type: none">- marcha : int- velocidade : float				
<ul style="list-style-type: none">+ mudarMarcha() : void+ mudarVelocidade() : void				

Onde quer que você olhe no mundo real, você vê objetos animados e inanimados.

Todos eles tem atributos (por exemplo, tamanho, forma, cor) e comportamentos;

Objeto	Comportamentos
	Uma bola gira, rebate, infla e murcha.

Objeto	Comportamentos
	<p>Um bebê mama, chora, dorme, engatinha, anda e pisca.</p>
	<p>Um carro acelera, freia e desvia.</p>
	<p>Uma toalha absorve água e seca.</p>

➔ ENCAPSULAMENTO:

Permite ignorar os detalhes de implementação permitindo ao desenvolvedor idealizar seu trabalho em um nível mais alto de abstração. Logo, o princípio do encapsulamento é a forma pela qual o programa é dividido a ponto de se tornar o mais isolado possível, ou seja, cada método pode ser executado isoladamente e retornar um resultado satisfatório ou não para cada situação. Sendo assim, o objeto não necessita conhecer qual forma cada método foi implementado.

O encapsulamento esconde a implementação interna da especificação externa:

- Clientes conhecem somente a interface;

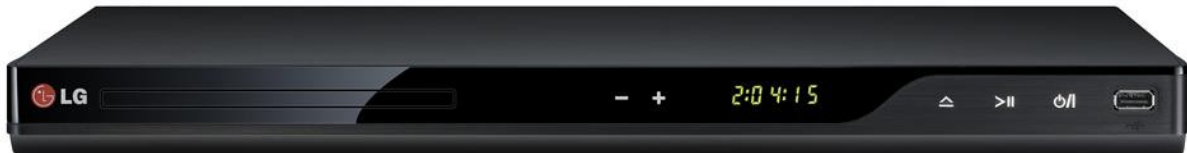
- Clientes dependem da interface e não da implementação.

O encapsulamento oferece os seguintes tipos de proteção:

- O estado interno de um objeto não pode ser corrompido por um cliente;
- Mudanças internas não têm impacto sobre os clientes.

Portanto, encapsulamento é o termo formal que descreve a junção de métodos e dados dentro de um objeto de maneira que acesso a estes dados só seja permitido por meio dos próprios métodos do objeto.

Ex: O DVD Player. Ninguém precisa conhecer detalhes dos circuitos de um DVD Player para utilizá-lo. Sua carcaça encapsula os detalhes e nos provê uma interface amigável.



Interface pública do DVD Player: voltar() pausar() avançar() parar() tocar()
carregarDisco() alterarHora()

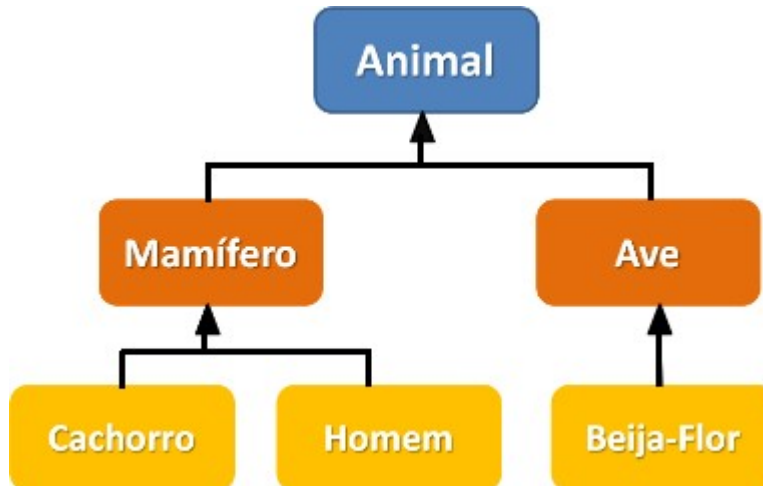
➔ HERANÇA:

Permite que elementos mais específicos incorporem a estrutura e o comportamento de elementos mais genéricos. Logo, nesse princípio diz que, uma classe pode compartilhar métodos e atributos entre si como, por exemplo, em um sistema escolar, onde temos uma classe Pessoa que contém características que definem uma pessoa. Porém, dentro do sistema temos uma outra classe Funcionário, que contém os mesmos atributos de Pessoa, além de outros atributos que apenas funcionários podem ter. Outro exemplo seria uma classe Aluno, que também contém atributos de pessoas e outros atributos que apenas pertencem a aluno.

A herança descreve o relacionamento entre classes definidas a partir de outras classes:

- Toda a subclasse herda os estados e os comportamentos definidos na superclasse;
- As subclasses não estão limitadas a estes estados e comportamentos, pois elas podem definir seus próprios atributos e comportamentos.

Exemplo:



O que é e o que não é herança:

O que é	O que não é
Um objeto de uma subclasse (classe filha) é um tipo de objeto da superclasse (classe pai).	Se um objeto de uma subclasse não possui todos os atributos e operações da superclasse, ela (classe deste objeto) não pode ser uma subclasse.
Exemplo: Uma mountain bike (subclasse) é uma bicicleta (superclasse).	Exemplo: Um gato não é um coelho.

→ POLIMORFISMO:

Permite que uma mesma operação possa ser definida para diferentes tipos de classes, e cada uma delas a implementa como quiser. Logo, polimorfismo é uma característica na qual os mesmos atributos ou métodos podem ser utilizados por objetos distintos e com implementações de lógica diferentes. Por exemplo, podemos dizer que um carro e uma bicicleta são veículos utilizados para locomoção e ambos contêm um método “Andar” em comum, porém, a implementação de cada um deles é feita de forma diferente.

Portanto, polimorfismo é o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma assinatura (lista de argumentos e tipo de retorno) mas com comportamentos distintos (especializados em cada subclasse):

- A decisão sobre qual método deve ser selecionado, de acordo com o tipo da classe;

- Permite a um mesmo objeto se manifestar de diferentes formas;
- Em linguagens fortemente tipadas o polimorfismo é implementado através de herança ou implementação de interfaces.

DESENVOLVIMENTO

AUTOMÓVEL (CARRO):

Um automóvel é um veículo usado para o transporte de pessoas.

- Substantivos: porta, roda, motor, chassi, marcha, embreagem, ignição, freio, tanque.
- Verbos: acelerar, frear, abrir porta, encher tanque, dar partida.

Objeto	Atributos	Comportamentos
Porta	Número, tipo de vidro, cor.	Abre, fecha, sobe vidro, desce vidro.
Roda	Tipo, tração.	Acelera, desacelera.
Ignição	Tipo automático, marca, modelo.	Dar partida.
Freio	Marca, modelo.	Freia o carro.
Motor	Marca, modelo, número de cavalos.	Combustão que se converte em movimento.

O motor queima o combustível que vem do tanque, os gases provenientes da combustão empurram o pistão, gerando o movimento que é transmitido para as rodas do carro.

LÂMPADA À VENDA NO SUPERMERCADO:

Classe Lampada

Atributos:

- Marca;
- Modelo;
- Estado (apagada, acesa e meia-luz);
- Potencia (em Watts);
- Status (Vendida / Prateleira).

Comportamentos:

- Testar: retorna o estado da lâmpada após ligar a mesma;
- Vender; vende para um cliente e atualiza o status para 'vendida'.

OBJETOS COM CARACTERÍSTICAS E COMPORTAMENTOS EM COMUM:

Objetos	Características e Comportamentos em Comum
Estudante, Cobra	Astutos
Avião, Carro	Transporte de pessoas
Avião, Borboleta, Pássaro	Voam
Lâmpada, Vela	Iluminam
Calculadora, Computador	Efetuem cálculos numéricos
Peixe, Baleia	Animais aquáticos
Cavalo, Coelho	Animais terrestres

CONCLUSÕES

A POO é realmente muito indicada para o desenvolvimento de sistemas atualmente, uma vez que permite a abstração do mundo real a partir da modelagem de objetos. Além do que, permite um ótimo reaproveitamento de código por herança e polimorfismo (indisponíveis na programação estruturada).

REFERÊNCIAS BIBLIOGRÁFICAS

1. OLIVEIRA, R. **POO: Motivação e Introdução**. IFG / Câmpus Goiânia. 2016.
2. SANTOS, I. **Programação Orientada a Objetos versus Programação Estruturada**. Disponível em: <http://www.devmedia.com.br/programacao-orientada-a-objetos-versus-programacao-estruturada/32813>. Acessado em: 18/09/2016.