Senai Governador Valadares CFP Luiz Chaves

Sistema POO no Desktop

Governador Valadares, janeiro de 2024

Alessandro Júnio

Arthur Prates

Átila Bragatto

Clarisse Oliveira

Gabriela Purger

Gabriel Pinto

Guilherme Augusto

Guilherme Duarte

Isadora Avelino

Júlio Cézar

Ronan Menezes

Leandro Telles

Marcos Vinicius

Miguel Salmen

Sistema POO no Desktop

Governador Valadares, janeiro de 2024

O que é Programação orientada a objetos (POO)

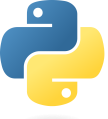
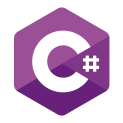
Programação orientada a objetos (POO, ou OOP segundo as suas siglas em inglês) é um paradigma de programação baseado no conceito de "objetos", que podem conter dados na forma de campos, também conhecidos como atributos, e códigos, na forma de procedimentos, também conhecidos como métodos.

Dentro do POO, é encontrado 4 pilares deste paradigma.

* Herança;
* Polimorfismo;
* Encapsulamento;
* Abstração.

Exemplos de POO:

Java, C++, C#, Python, PHP e Ruby.

Encapsulamento no POO em Desktop

O que é?

Em sistemas de software, o encapsulamento refere-se ao agrupamento de dados com os mecanismos ou métodos que operam sobre os dados. Também pode referir-se à limitação do acesso direto a alguns desses dados, como os componentes de um objeto. Essencialmente, o encapsulamento evita que o código externo se preocupe com o funcionamento interno de um objeto.

No POO em Desktop

Todos os sistemas de programação orientada a objetos (POO) suportam encapsulamento, mas o encapsulamento não é exclusivo do POO. Implementações de tipos de dados abstratos, módulos e bibliotecas, entre outros sistemas, também oferecem encapsulamento. A semelhança foi explicada pelos teóricos das linguagens de programação em termos de tipos existenciais.

O encapsulamento permite que os desenvolvedores apresentem uma interface consistente e independente de sua implementação interna. Como exemplo, o encapsulamento pode ser usado para ocultar os valores ou o estado de um objeto de dados estruturados dentro de uma classe, impedindo o acesso direto a eles pelos clientes de uma forma que poderia expor detalhes ocultos de implementação ou violar a invariância de estado mantida pelos métodos.

Também incentiva os programadores a colocarem todo o código relacionado a um determinado conjunto de dados na mesma classe, o que o organiza para fácil compreensão por outros programadores. O encapsulamento é uma técnica que incentiva o desacoplamento.

Os modificadores de acesso mais comuns são:

* private — os atributos e os métodos marcados como “private” só podem ser acessados dentro da própria classe em que foram declarados;
* protected — os atributos e os métodos marcados como “protected” podem ser acessados dentro da própria classe e das subclasses;
* public — os atributos e os métodos marcados como “public” podem ser acessados por qualquer classe.

Algumas [linguagens de programação](https://blog.locaweb.com.br/temas/codigo-aberto/linguagens-de-programacao-web/) têm outros modificadores, como “internal” e “package-private”, que variam de acordo com a linguagem, mas em geral têm a mesma funcionalidade de proteção de acesso. É importante notar que o uso correto desses modificadores de acesso é fundamental para garantir a segurança e a integridade do código.

O que pode ser encapsulado?

O encapsulamento pode ser aplicado em várias linguagens de programação. Em geral, a maioria das linguagens orientadas a objetos, como Java, C#, C++, Python, Ruby, JavaScript e PHP, tem mecanismos para encapsulamento, mesmo que as sintaxes e as nomenclaturas possam variar.

Além das linguagens orientadas a objetos, existem outros exemplos de encapsulamento em diferentes áreas, como:

* rede — a ferramenta adiciona cabeçalhos e trailers a pacotes de dados para permitir o transporte de dados por diferentes tipos de rede;
* microcontroladores — usados para criar uma camada de abstração entre o hardware e o software, permitindo que o software acesse o hardware de forma simples e intuitiva;
* bibliotecas, componentes e serviços — escondem os detalhes de implementação dessas funções e procedimentos, expondo somente a interface pública.

Herança

Herança é um dos pontos chave de programação orientada a objetos (POO). A ideia de herança é facilitar a programação. Uma classe A deve herdar de uma classe B quando podemos dizer que A é um B. Por exemplo, imagine que já exista uma classe que defina o comportamento de um dado objeto da vida real, por exemplo, animal.

A Herança possibilita que as classes compartilhem seus atributos, métodos e outros membros da classe entre si. Para a ligação entre as classes, a herança adota um relacionamento esquematizado hierarquicamente.

Na Herança temos dois tipos principais de classe:

**Classe Base:** A classe que concede as características a uma outra classe.

**Classe Derivada:** A classe que herda as características da classe base.

O fato de as classes derivadas herdarem atributos das classes bases assegura que programas orientados a objetos cresçam de forma linear e não geometricamente em complexidade. Cada nova classe derivada não possui interações imprevisíveis em relação ao restante do código do sistema.

Com o uso da herança, uma classe derivada geralmente é uma implementação especifica de um caso mais geral. A classe derivada deve apenas definir as características que a tornam única.

Polimorfismo

O polimorfismo consiste em um mesmo método para apresentar comportamentos diferentes, dependendo da classe em que seja chamado. É um mecanismo por meio do qual selecionamos as funcionalidades utilizadas de forma dinâmica por um programa no decorrer de sua execução.

Ele pode apresentar duas classificações:

**Polimorfismo estático**: ocorre em momento de compilação. O mesmo método é implementado várias vezes na mesma classe, com parâmetros diferentes. A escolha do método a ser chamado vai variar de acordo com o parâmetro passado. O polimorfismo estático também pode ser chamado de sobrecarga, enquanto o polimorfismo dinâmico também pode ser chamado de sobreposição.

**Polimorfismo dinâmico**: ocorre em momento de execução. O mesmo método é implementado várias vezes nas subclasses derivadas, com os mesmos parâmetros. A escolha do método depende do objeto que o chama (e, consequentemente, da classe que o implementa).

Com o Polimorfismo, os mesmos atributos e objetos podem ser utilizados em objetos distintos, porém, com implementações lógicas diferentes. Por exemplo: podemos assumir que uma bola de futebol e uma camisa da seleção brasileira são artigos esportivos, mas que o cálculo deles em uma venda é calculado de formas diferentes.

Outro exemplo: podemos dizer que uma classe chamada Vendedor e outra chamada Diretor podem ter como base uma classe chamada Pessoa, com um método chamado CalcularVendas. Se este método (definido na classe base) se comportar de maneira diferente para as chamadas feitas a partir de uma instância de Vendedor e para as chamadas feitas a partir de uma instância de Diretor, ele será considerado um método polimórfico, ou seja, um método de várias formas.

Outros princípios fundamentais da POO

1. A abstração na POO é a capacidade de ocultar detalhes irrelevantes ou complexos de um problema e focar nos aspectos essenciais. Ela é implementada por meio de classes, que são modelos que descrevem os atributos e comportamentos comuns de um grupo de objetos.
2. Associação e herança são dois dos principais conceitos da programação orientada a objetos (POO). Associação é um relacionamento entre duas ou mais classes, em que uma classe usa objetos de outra classe como parte de suas operações.
3. Composição ou Agregação é um mecanismo de reaproveitamento (reutilização) de classes utilizado pela POO para aumentar a produtividade e a qualidade no desenvolvimento de software. Reaproveitamento ou reutilização de classes significa que você pode usar uma ou várias classes para compor outra classe.
4. Composição é um tipo mais forte de agregação, onde as partes são partes essenciais do todo e não podem existir independentemente. Se o todo for destruído, suas partes também serão destruídas.

Casos de uso específico

As aplicações desktop são normalmente usadas para tarefas que exigem mais poder de processamento, segurança, estabilidade ou personalização do que as aplicações web. Alguns exemplos de aplicações desktop são:

**Aplicações de escritório:** processadores de texto, planilhas, editores de texto, etc. Essas aplicações permitem criar, editar e gerenciar documentos, dados e arquivos de forma rápida e eficiente.

**Sistemas de negócios:** sistemas de gestão, contabilidade, estoque, vendas, etc. Essas aplicações facilitam o controle e a organização das operações e dos recursos de uma empresa.

**Utilitários de administração dos dados do computador:** antivírus, backup, limpeza, otimização, etc. Essas aplicações ajudam a proteger, recuperar, limpar e melhorar o desempenho do computador.

**Jogos:** simuladores, aventuras, estratégia, ação, etc. Esses aplicativos proporcionam entretenimento, diversão e desafio aos usuários, explorando gráficos, sons, interatividade e inteligência artificial.

**Softwares educativos:** cursos, tutoriais, exercícios, etc. Esses aplicativos auxiliam no aprendizado, no ensino e na avaliação de conteúdos educacionais.

**Aplicações específicas:** editores de vídeo, de imagem, de áudio, etc. Essas aplicações atendem a necessidades particulares de usuários que trabalham com mídias, artes, design, etc...

Como foi criado o POO?

Em busca de uma melhora no deisgn de software, o paradigma de programação POO foi criado com o intuito de tornar tais designs mais modulares, fáceis de manter e escaláveis.

Ela tem sua origem no entre o final dos anos 60 e começo do 70, porém foi realmente introduzido no final dos anos 80.

Pensado principalmente por Alan Kay (com a ajuda de Bertrand Meyer e Grady Booch), a ideia por trás da POO é, como o nome diz, usar de objetos do mundo real e seus comportamentos em um programa de computador; fazendo assim com que o entendimento sobre POO e toda sua estrutura seja muito mais fácil de entender.

Pioneiros da POO

O termo Programação Orientada a Objetos foi criado por Alan Kay, autor da linguagem de programação Amalltalk. Mas mesmo antes da criação do Smalltalk, algumas das ideias da POO já eram aplicadas, senado que a primeira linguagem a realmente utilizar estas ideias foi a linguagem Simula 67, criada por Ole Joah Dahl e Kristen Nygard em vem sendo aceito realmente nas grandes empresas de desenvolvimento de Software. Alguns exemplos de linguagens modernas utilizadas por grandes empresas em toso o mundo que adotaram essas ideias: Java,C++,Object Pascal (Delphi), Ruby, Python, Lisp, etc.

Exemplos de linguagem

Exemplos de linguagens de programação desktop:

1- Python: Python é uma linguagem de programação de alto nível, interpretada de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi lançada por Guido van Rossum em 1991. Atualmente, possui um modelo de desenvolvimento comunitário, aberto e gerenciado pela organização sem fins lucrativos Python Software Foundation. Apesar de várias partes da linguagem possuírem padrões e especificações formais, a linguagem, como um todo, não é formalmente especificada. O padrão na pratica é a implementação CPython.

2- Embarcadero Delphi: Embarcadero Delphi, anteriormente conhecido como CodeGear Delphi, Inprise Delphi e Borland Delphi, também conhecido como Delphi, é um compilador, uma IDE e uma linguagem de programação, produzido anteriormente pela Borland Software Corporation e atualmente pela Embarcadero.[1] O Delphi, originalmente direcionado para a plataforma Windows, chegou a ser usado para desenvolvimento de aplicações nativas para Linux através do Kylix (o Kylix é um IDE para as linguagens C++ e Object Pascal), e para o framework Microsoft .NET em suas versões mais recentes. O desenvolvimento do Kylix foi descontinuado.

3- C#: O C# é uma linguagem de programação multiparadigma criada pela Microsoft, sendo a principal da plataforma .NET. Por ser uma linguagem que suporta, entre outros paradigmas, a orientação à objetos, ela suporta conceitos comuns como encapsulamento, herança e polimorfismo. Trata-se também de uma linguagem fortemente tipada e case-sensitive, ou seja, faz diferenciação entre letras minúsculas e maiúsculas.

Onde se aplica

Para a aplicação desktop, podemos pensar em uma aplicação que é instalada e, em sua maioria, roda no computador desktop ou laptop. Deve-se notar que as aplicações desktop também podem ser executadas em dispositivos portáteis como telefones, tablets e outros dispositivos móveis que suportam a estrutura .NET, porém isto está fora do escopo deste artigo.

As aplicações de desktop são muito poderosas porque oferecem mais controle e estabilidade do que as aplicações baseadas na web e até mesmo mobiles. Ela pode ser usada em diversas situações, como:

-Sistemas de gestão de estoque.

-Ferramentas de contabilidade e finanças.

-Sistemas de gerenciamento de recursos humanos (RH).

Onde se pode usar a Microsoft Excel ou usar ferramentas de análise de dado em Matlab.

Essas são as várias formas e situações onde se pode realizar a aplicação do desktop.

Quais os melhores cenários em comparação à programa desktop e aplicação web?

Principais diferenças entre Aplicações Desktop e Aplicações Web

A principal diferença entre aplicações desktop e aplicações baseadas na web é que as aplicações desktop são offline por padrão, enquanto as aplicações baseadas na web são online por padrão.

As aplicações desktop, software projetado para computadores desktop (como Windows ou Mac), funcionam em um computador de cada vez. Não está disponível para mais ninguém, a menos que o usuário o envie por e-mail ou por um drive externo. Isto inclui a configuração de configurações específicas para aquele computador, como fundos de desktop e protetores de tela. Ao salvar arquivos, os aplicativos de área de trabalho só salvarão dados localmente (naquele computador).

As aplicações baseadas na Web podem ser acessadas através de qualquer tipo de conexão com a Internet; elas são executadas através de navegadores da Web, como o Google Chrome ou o Microsoft Edge, tanto em PCs desktop quanto em dispositivos móveis. Os dados são sempre salvos em servidores remotos (sobre os quais você não tem controle) para que possam ser recuperados mesmo quando você não tem acesso para a versão desktop do programa.



No caso da segurança de dados

No caso do sistema desktop é bem simples, ele vai guardar tudo no seu computador, juntamente com o próprio software. Mas aconselhamos sempre realizar o seu backup em uma mídia externa como Pendrive ou HD externo e retirar da sua máquina após o procedimento, para que eles também não sejam invadidos por ataques de vírus como o Ransomware.

No entanto, o sistema Web funciona de maneira diferente. Todos os dados cadastrados são enviados para o servidor em nuvem onde seu sistema está, e nele serão armazenado.

Principais vantagens:

1. O sistema desktop não depende de internet;

Dependendo apenas do seu computador, estando online ou offline, você consegue usá-lo da mesma maneira, mas claro com as mudanças das legislações de cada estado as documentações fiscais (NF-eNFC-e) precisam de internet para serem transmitidas.

### Desempenho:

### Programas desktop têm acesso direto ao hardware do dispositivo, o que geralmente resulta em um desempenho mais rápido e eficiente em comparação com aplicativos web ou móveis.

### Personalização:

Os aplicativos desktop podem ser altamente personalizados de acordo com as necessidades do usuário ou da empresa. É possível criar interfaces intuitivas e adaptadas aos fluxos de trabalho específicos, proporcionando uma experiência mais confortável e familiar.

1. Segurança:

O desenvolvimento de software desktop oferece um maior controle sobre a segurança dos dados. Os aplicativos podem ser projetados para armazenar informações localmente, reduzindo os riscos associados ao compartilhamento de dados pela internet.

### Integração com o sistema operacional:

Os aplicativos desktop podem se integrar perfeitamente com o sistema operacional do usuário, aproveitando recursos nativos e fornecendo uma interação mais fluida. Isso permite a criação de funcionalidades avançadas e personalizadas que não são possíveis com aplicativos web ou móveis.

Desvantagens:

1. Uma interface gráfica muito carregada deixa a aplicação mais pesada
2. A integração com usuários remotos é mais sofrida
3. A distribuição da aplicação é crítica. Você tem que instalar sua aplicação em diferentes tipos de máquinas, com diferentes tipos de sistemas, drives e periféricos.
4. A manutenção e atualização de sua aplicação requerem um esforço extra.

#### Sistema Web

Principais vantagens:

1. Interface HTML reconhecida por uma grande gama de usuários já acostumados com o funcionamento dos navegadores;
2. Desenvolvimento, manutenção e atualização centralizada da aplicação. Você não precisa sair instalando sua aplicação em diversos equipamentos diferentes. Basta colocá-lo no servidor para que os usuários a acessem;
3. A exportação de dados entre usuários remotos usando o protocolo HTTP é muito mais fácil do que usar outro protocolo.
4. Se houver necessidade de aumentar o poder de processamento, basta fazer isto no servidor.

Desvantagens:

1. A interface HTML pode ser um problema, pois não há uma padronização entre os diversos navegadores e sua aplicação poderia ser exibida de uma maneira diferente dependendo do navegador
2. A entrada de uma grande massa de dados é prejudicada na interface HTML, pois não existe uma maneira padrão de criar máscaras de entrada de dados.
3. A interface HTML não é rica em controles gráficos e peca no quesito posicionamento. O visual da sua aplicação pode não ficar tão elegante como você imagina.
4. A integração com outros componentes não é tão fácil com HTML.

Referências

* O que é Programação orientada a objetos (POO):

https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o\_orientada\_a\_objetos

https://coodesh.com/blog/dicionario/o-que-e-poo-em-programacao/

* Encapsulamento no POO em Desktop:

<https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o_orientada_a_objetos#Encapsulamento>

<https://www.locaweb.com.br/blog/temas/codigo-aberto/o-que-e-encapsulamento/>

* Herança:

https://www.devmedia.com.br/

* Polimofismo:

<https://www.estrategiaconcursos.com.br/blog/heranca-polimorfismo-programacao-orientada-objetos/>

* Outros princípios:

<https://www.dio.me/articles/pilares-de-poo-em-java#:~:text=A%20abstra%C3%A7%C3%A3o%20na%20POO%20%C3%A9,de%20um%20grupo%20de%20objetos>.

<https://www.dio.me/articles/programacao-orientada-a-objetos-heranca-x-associacao#:~:text=Associa%C3%A7%C3%A3o%20e%20heran%C3%A7a%20s%C3%A3o%20dois,como%20parte%20de%20suas%20opera%C3%A7%C3%B5es>.

<https://materialpublic.imd.ufrn.br/curso/disciplina/5/8/6/2#:~:text=Composi%C3%A7%C3%A3o%20ou%20Agrega%C3%A7%C3%A3o%20%C3%A9%20um,classes%20para%20compor%20outra%20classe>.

* Casos de uso específico:

https://wktechnology.com.br/aplicacoes-desktop-o-que-sao-e-como-funcionam/

<https://pt.stackoverflow.com/questions/187344/quais-são-as-diferenças-entre-uma-aplicação-web-e-a-aplicação-desktop>

* Como foi criado o POO:

https://youtu.be/KlIL63MeyMY?si=cIunkgGPPMBDEUwp

https://pt.wikipedia.org/wiki/Programa%C3%A7%C3%A3o\_orientada\_a\_objetos

* Pionerios:

<https://www.hardwre.com.br/artigos/programacao-orientada-objetos/>

* Exemplos de linguagem:

https://pt.wikipedia.org/wiki/Pytho

https://pt.wikipedia.org/wiki/Delphi\_(software)

https://www.treinaweb.com.br/blog/o-que-e-e-como-comecar-com-c-sharp

* Onde se aplica:

<https://wktechnology.com.br/aplicacoes-desktop-o-que-sao-e-como-funcionam/#:~:text=Para%20a%20aplica%C3%A7%C3%A3o%20desktop%2C%20podemos,m%C3%B3veis%20que%20suportam%20a%20estrutura%20>.

* Quais os melhores cenários em comparação à programa desktop e aplicação web:

<https://compucenter.net.br/sistema-web-ou-desktop-qual-melhor-opcao-para-minha-empresa/#:~:text=Aplica%C3%A7%C3%A3o%20web%20%C3%A9%20aquela%20acessada,como%20Windows%2C%20Linux%20ou%20MacOS.>

<https://wktechnology.com.br/aplicacoes-desktop-o-que-sao-e-como-funcionam/#:~:text=Uma%20aplica%C3%A7%C3%A3o%20desktop%20%C3%A9%20um,pela%20maioria%20das%20aplica%C3%A7%C3%B5es%20deskto>p.

<https://awari.com.br/desenvolvimento-de-software-desktop-guias-e-melhores-praticas/?utm_source=blog&utm_campaign=projeto+blog&utm_medium=Desenvolvimento%20de%20software%20desktop:%20Guias%20e%20melhores%20pr%C3%A1ticas#:~:text=Um%20dos%20principais%20benef%C3%ADcios%20do,a%20produtividade%20e%20a%20flexibilidade.>

<https://bitmasters.wordpress.com/2011/03/03/sistema-web-x-sistema-desktop/>