Escola João XXIII – Jardim São Jorge

Christopher Apariz Da Conceição

3°B

**Programação Orientada a Objetos:**

Explorando os Fundamentos e Vantagens da POO

São Paulo

22/04/24

Christopher Apariz Da Conceição

**Programação Orientada a Objetos:**

Explorando os Fundamentos e Vantagens da POO

Descubra os princípios da Programação Orientada a Objetos POO e suas aplicações em Java, C# e Python. Este Trabalho mostra os conceitos fundamentais e fornece exemplos práticos para uma compreensão sólida.

Orientador: Prof - Gustavo

São Paulo

22/04/24

Introdução:

A programação orientada a objetos (POO) marcou uma revolução nas décadas de 1960 e 1970, com linguagens como Simula e Smalltalk.

Essa abordagem inovadora introduziu conceitos como classes, objetos, herança e polimorfismo, transformando profundamente a maneira como os programas são desenvolvidos.

Linguagens como C + +, Java e Python impulsionaram ainda mais a POO, tornando-a essencial para o desenvolvimento de software modular, reutilizável e fácil de manter.

Desde então, a POO se consolidou como um paradigma dominante, moldando a forma como sistemas complexos são projetados e implementados.

Como foi o surgimento da programação orientada a objetos:

O surgimento da programação orientada a objetos foi um processo gradual, impulsionado por uma combinação de avanços técnicos, teóricos e práticos.

Essa abordagem revolucionária transformou a maneira como os programas são desenvolvidos e continua a ser uma das abordagens mais populares e poderosas para o desenvolvimento de software até hoje.

Quais os princípios da programação orientada a objetos

Abstração: é um conceito que permite criar modelos simplificados de entidades complexas do mundo real. Ao focar nos aspectos mais importantes para a aplicação em desenvolvimento, a abstração ajuda a simplificar a representação de objetos e conceitos, tornando mais fácil o desenvolvimento e a compreensão do sistema.

Encapsulamento: é um princípio da programação orientada a objetos que visa proteger os dados de um objeto, evitando modificações inadequadas. Ele funciona ocultando os detalhes internos de um objeto que não são relevantes para seu uso externo, permitindo apenas o acesso controlado aos seus atributos e métodos.

Herança: é um conceito fundamental na programação orientada a objetos, que permite que classes compartilhem atributos e métodos entre si. Isso significa que uma classe pode herdar características de outra classe, permitindo o reuso de código e a especialização de operações ou atributos.

Polimorfismo: é um conceito chave na programação orientada a objetos, que permite que o mesmo elemento seja usado de maneiras diferentes. Isso significa que um objeto pode ser tratado de várias formas, dependendo do contexto em que é utilizado.

Quais os conceitos fundamentais da POO.

Classes e Objetos: As classes são modelos para a criação de objetos, enquanto os objetos são instâncias específicas de uma classe. As classes definem os atributos e comportamentos comuns a um conjunto de objetos, enquanto os objetos são instâncias individuais que possuem valores específicos para esses atributos.

Encapsulamento: É o princípio de esconder os detalhes de implementação de um objeto e fornecer uma interface clara para interagir com ele. Isso é alcançado definindo membros privados dentro de uma classe e fornecendo métodos públicos para acessar e modificar esses membros.

Herança: Permite que uma classe herde atributos e métodos de outra classe. A classe que herda é chamada de subclasse ou classe filha, e a classe da qual ela herda é chamada de superclasse ou classe pai. Isso promove a reutilização de código e estabelece uma hierarquia entre as classes.

Polimorfismo: Refere-se à capacidade de objetos de diferentes classes serem tratados de maneira uniforme. Isso é alcançado através da implementação de métodos com o mesmo nome em classes diferentes, mas com comportamentos específicos para cada classe. O polimorfismo permite que um método seja chamado de maneira genérica, mas a execução específica depende do tipo do objeto no momento da chamada.

Abstração: Envolve identificar os aspectos essenciais de um objeto e ocultar os detalhes irrelevantes. A abstração permite representar objetos do mundo real de uma forma simplificada e concentrar-se apenas nos aspectos relevantes para o sistema em questão.

Vantagens da programação orientada a objetos.

Reutilização de código: O POO permite a criação de classes e objetos que podem ser reutilizados em diferentes partes do programa ou mesmo em projetos diferentes. Isso economiza tempo e esforço, pois o código já existente pode ser aproveitado em vez de ser reescrito.

Encapsulamento: O encapsulamento permite ocultar os detalhes de implementação de uma classe e expor apenas uma interface pública. Isso protege os dados e comportamentos de uma classe, promovendo a segurança e a integridade dos objetos.

Herança: A herança permite que uma classe herde atributos e métodos de outra classe, promovendo a reutilização de código e a extensibilidade do sistema. Isso facilita a criação de hierarquias de classes e promove uma estrutura mais organizada e coesa.

Polimorfismo: O polimorfismo permite que objetos de diferentes classes sejam tratados de maneira uniforme. Isso promove a flexibilidade e a extensibilidade do sistema, permitindo que novas classes sejam adicionadas sem alterar o código existente.

Organização modular: A POO promove a organização do código em unidades modulares, conhecidas como classes, que representam entidades do mundo real. Isso facilita a manutenção do código, pois cada classe é responsável por uma única funcionalidade ou conjunto de funcionalidades.

Desenvolvimento colaborativo: A POO facilita o desenvolvimento colaborativo, permitindo que diferentes desenvolvedores trabalhem em classes ou módulos específicos do sistema. Isso promove a divisão de tarefas e aumenta a produtividade da equipe.

Modelagem realista: O POO permite modelar sistemas de software de forma mais próxima à realidade, usando objetos para representar entidades do mundo real. Isso facilita a compreensão do sistema por parte dos desenvolvedores e stakeholders, tornando o código mais intuitivo e fácil de dar manutenção.

Comparação com outros paradigmas de programação.

Java: Desenvolvida pela Sun Microsystems, tornou-se um símbolo da POO e é conhecida por sua portabilidade e uso extensivo de classes e objetos.

C#: Desenvolvida pela Microsoft como parte da plataforma. NET, é semelhante a Java em estrutura e sintaxe, mas oferece recursos adicionais para desenvolvimento de componentes de software.

Python: Linguagem de programação de alto nível, conhecida pela sua simplicidade e versatilidade, permite o desenvolvimento tanto em programação estruturada quanto em POO.

A POO: permite a reutilização de código e a modelagem de problemas de forma modular, usando classes, objetos, herança e polimorfismo, fornecendo suporte para tipos de dados abstratos, herança e vinculação dinâmica de chamadas a métodos.

A POO oferece uma abordagem flexível e modular para o desenvolvimento de software, permitindo uma melhor organização e manutenção do código em projetos de grande escala.

Comparativamente, outros paradigmas de programação como o imperativo, funcional e lógico têm suas próprias características e são mais adequados para certos tipos de problemas ou contextos de desenvolvimento.

Conclusão:

A programação orientada a objetos é um paradigma poderoso que transformou a maneira como os programas são desenvolvidos.

Com seus princípios fundamentais, conceitos e vantagens distintas, a POO oferece uma abordagem eficiente e intuitiva para modelar problemas do mundo real.

À medida que continuamos a explorar novas tecnologias e desafios de desenvolvimento de software, a POO continuará desempenhando um papel fundamental no progresso da computação.

Fontes:

<https://www.dio.me/articles/explorando-os-fundamentos-da-programacao-orientada-a-objetos>

<https://remsoft.com.br/blog/tecnologias/surgimento-da-orientacao-a-objetos/#:~:text=A%20orienta%C3%A7%C3%A3o%20a%20objetos%20tem,conceitos%20de%20classe%20e%20heran%C3%A7a>.

<https://www.alura.com.br/artigos/poo-programacao-orientada-a-objetos#:~:text=farol%20do%20carro-,Encapsulamento%2C%20heran%C3%A7a%20e%20polimorfismo%3A%20as%20principais%20caracter%C3%ADsticas%20da%20POO,conceitos%20de%20classe%20e%20objeto>

<https://www.dio.me/articles/vantagens-da-programacao-orientada-a-objetos-poo>

<https://tripleten.com.br/blog/paradigmas-de-programacao-o-que-sao-e-quais-os-principais/>

<https://www.devmedia.com.br/entendendo-e-aplicando-heranca-em-java/24544>

<https://www.devmedia.com.br/os-4-pilares-da-programacao-orientada-a-objetos/9264>

<https://www.devmedia.com.br/como-criar-minha-primeira-classe-em-csharp/38785>

Bibliografia:

Gasparotto, H. M. Os 4 pilares da Programação Orientada a Objetos. DevMedia, Rio de Janeiro, 2014.

Geovane, H. Entendendo e Aplicando Herança em Java. DevMedia, Rio de Janeiro, 2012.

Ledur, C. L. Desenvolvimento de sistemas com C#. Porto Alegre: SAGAH, 2018.

Lutz, M.; Ascher, D. Aprendendo Python. 2. ed. Porto Alegre: Bookman; O’Reilly, 2007.

Machado, R. P.; Franco, M. H. I.; Bertagnolli, S. C. Desenvolvimento de software III: programação de sistemas web orientada a objetos em Java. Porto Alegre: Bookman, 2016.

Rodrigues, J. Como criar minha primeira classe em C#. DevMedia, Rio de Janeiro, 2017.

Sebesta, R. W. Conceitos de linguagem de programação. 11. ed. Porto Alegre: Bookman, 2018.

Tucker, A. B.; Noonan, R. E. Linguagens de programação: princípios e paradigmas. 2. ed. Porto Alegre: AMGH, 2009.