Professor Gustavo – Linguagem a objetos 10/10/2024

Rust é uma linguagem de programação multiparadigma e compilada, desenvolvida pela Mozilla Research, que se propõe a ser segura, prática e concorrente.

Rust é uma linguagem de sistemas, utilizada para escrever sistemas como sistemas operacionais. É inspirada por outras linguagens, como C++, C# e Haskell.

Alguns dos principais princípios da Rust são: Concorrência sem disputa de dados, Abstração sem overhead, Segurança sem coletor de lixo.

Rust é rápida e gerencia memória de forma eficiente. Alguns dos recursos da linguagem são: Programação simultânea segura, Sistema de tipo expressivo, Verificações durante a compilação, Framework de testes integrado, Aplicação do tratamento de erros.

Rust pode ser utilizada em vários tipos de aplicações, como: Ferramentas de linhas de comando, Ferramentas para DevOps, Web services, Jogos, Aplicativos de IoT (Internet of Things).

Empresas automobilísticas e aeroespaciais já estão adotando Rust para criar código. A Microsoft também sugeriu que a empresa usará cada vez mais Rust para códigos novos.

 Prolog (Programação Lógica) é uma linguagem de programação que se enquadra no paradigma de Programação em Lógica Matemática. É uma linguagem de uso geral, mas é especialmente associada com a inteligência artificial e linguística computacional.

O Prolog foi criado para criar programas que traduzam linguagens naturais, como o português e o inglês.

Os elementos básicos da linguagem Prolog são: Fatos, Regras, Consultas.

O Prolog é diferente das linguagens de programação mais comuns, pois trata todos os dados como sendo de um único tipo, Termo. A natureza do termo depende da forma como foi declarado.

Algumas implementações do Prolog são: Turbo Prolog, Visual Prolog, ARITY Prolog, SWI Prolog.

O comando cut é usado para indicar ao Prolog quais sub-objetivos já satisfeitos não precisam ser reconsiderados. Isso permite que o programa rode mais rápido e ocupe menos memória

O paradigma procedural é um paradigma de programação que agrupa instruções em procedimentos, que são executados em sequência. É uma subcategoria da programação imperativa, e é muitas vezes usado como sinônimo dela.

Algumas características do paradigma procedural são:

* Estrutura: O programa é estruturado em torno de funções e procedimentos.
* Reutilização de código: O código é reutilizado através da criação de funções genéricas.
* Variáveis: As variáveis têm escopo global ou local e são passadas como parâmetros para as funções.
* Sequência de execução: As instruções são executadas na ordem em que aparecem, de cima para baixo.
* Estruturas de controle: Loops (for, while) e condicionais (if-else) ajudam a controlar o fluxo do programa.

O paradigma procedural é uma boa escolha para programas de média complexidade que precisam ser mantidos com facilidade.

Alguns exemplos de linguagens de programação procedurais são C, Pascal e Fortran.

O paradigma funcional é uma metodologia de programação que consiste em mapear valores de entrada em valores de retorno através de funções. É baseado no cálculo lambda, um modelo de computação criado em 1930 por Alonzo Church.

Algumas características do paradigma funcional são:

* Foco na avaliação de expressões e aplicação de funções matemáticas
* Uso de funções puras e imutáveis
* Evitar o uso de estados mutáveis e dados compartilhados
* Priorizar a expressividade e evitar efeitos colaterais

Algumas linguagens de programação que suportam o paradigma funcional são:

Haskell, Erlang, Scala, Clojure, OCaml, F#.

A programação funcional difere da programação orientada a objetos (POO) na forma como o código é organizado. A POO é uma programação imperativa, enquanto a programação funcional é uma programação declarativa

**Paradigma Lógico: Uma Abordagem Baseada na Lógica Matemática**

O **paradigma lógico** é um modelo de programação que se baseia na lógica matemática para resolver problemas. Ao invés de seguir uma sequência de instruções imperativas, como em linguagens procedurais, o paradigma lógico se concentra em **declarar fatos e regras** sobre um domínio específico, deixando que a máquina deduza as respostas a partir dessas informações.

**Como funciona?**

* **Fatos:** São declarações sobre o mundo que são assumidas como verdadeiras. Por exemplo: "Socrates é um homem", "Todos os homens são mortais".
* **Regras:** São inferências lógicas que permitem deduzir novas informações a partir dos fatos. Por exemplo: "Se x é um homem, então x é mortal".
* **Consultas:** São perguntas que o programador faz ao sistema, e que o sistema tenta responder com base nos fatos e regras. Por exemplo: "Socrates é mortal?".

**Exemplo simples:**

Prolog

homem(socrates).

mortal(X) :- homem(X).

?- mortal(socrates).

* **homem(socrates):** Declara que Sócrates é um homem.
* **mortal(X) :- homem(X):** Regra que diz que se X é um homem, então X é mortal.
* **?- mortal(socrates):** Consulta que pergunta se Sócrates é mortal.

O sistema Prolog, por exemplo, irá inferir que Sócrates é mortal com base nas regras e fatos declarados.

**Aplicações do Paradigma Lógico**

* **Inteligência Artificial:** Criação de sistemas especialistas, processamento de linguagem natural, e resolução de problemas.
* **Bases de dados:** Representação de conhecimento e realização de consultas complexas.
* **Prova de teoremas:** Verificação de teoremas matemáticos.

**Vantagens**

* **Expressividade:** Permite representar conhecimento de forma natural e intuitiva.
* **Modularidade:** Facilita a adição de novos fatos e regras ao sistema.
* **Declaração, não programação:** O programador declara o que sabe, e não como resolver o problema.

**Desvantagens**

* **Desempenho:** Pode ser menos eficiente que outros paradigmas para certos tipos de problemas.
* **Curva de aprendizado:** Requer um entendimento de lógica matemática.

**Linguagens Lógicas**

* **Prolog:** A linguagem lógica mais conhecida e utilizada.
* **Datalog:** Uma variação de Prolog com foco em bases de dados.

**Em resumo,** o paradigma lógico oferece uma abordagem poderosa e expressiva para resolver problemas que envolvem raciocínio lógico. É especialmente útil em áreas como inteligência artificial e representação de conhecimento.

**Gostaria de explorar algum aspecto do paradigma lógico em mais detalhes?** Por exemplo, podemos discutir:

* **Comparação com outros paradigmas de programação**
* **Aplicações mais específicas do paradigma lógico**
* **Conceitos mais avançados de programação lógica**
* **Linguagens lógicas além do Prolog**

**Lua: Uma Linguagem de Programação Versátil e Popular**

**O que é Lua?**

Lua é uma linguagem de programação interpretada, de alto nível, com tipagem dinâmica e multiparadigma, projetada para ser leve e extensível. Criada no Brasil, na PUC-Rio, Lua se tornou uma das linguagens de script mais populares do mundo, sendo utilizada em diversas áreas, desde jogos até sistemas embarcados.

**Características Principais:**

* **Leve e rápida:** Lua é conhecida por seu pequeno tamanho e alta performance, o que a torna ideal para aplicações com recursos limitados.
* **Extensível:** Lua permite a criação de módulos personalizados, o que facilita a adaptação da linguagem a diferentes projetos.
* **Embutida:** Lua pode ser facilmente integrada a outras linguagens e plataformas, tornando-a uma excelente opção para desenvolvimento de jogos e aplicativos.
* **Sintaxe simples:** A sintaxe de Lua é clara e concisa, facilitando o aprendizado e a escrita de código.
* **Orientada a tabelas:** Lua utiliza tabelas associativas como estrutura de dados principal, proporcionando uma forma flexível e eficiente de armazenar informações.

**Para que serve Lua?**

Lua é utilizada em diversas áreas, como:

* **Desenvolvimento de jogos:** Lua é amplamente utilizada em jogos para criar scripts de gameplay, interfaces e ferramentas de desenvolvimento.
* **Sistemas embarcados:** Lua é utilizada em dispositivos com recursos limitados, como roteadores, dispositivos IoT e sistemas de controle industrial.
* **Prototipagem:** Lua é uma ótima opção para criar protótipos rapidamente, graças à sua sintaxe simples e à facilidade de desenvolvimento.
* **Aplicações científicas:** Lua é utilizada em diversas áreas da ciência, como simulação e análise de dados.

**Por que usar Lua?**

* **Facilidade de aprendizado:** Lua possui uma curva de aprendizado suave, tornando-a acessível a programadores de todos os níveis.
* **Grande comunidade:** Lua possui uma comunidade ativa e engajada, oferecendo diversos recursos e bibliotecas.
* **Portabilidade:** Lua pode ser executada em diversas plataformas, desde sistemas operacionais de desktop até dispositivos móveis.
* **Flexibilidade:** Lua permite uma grande variedade de estilos de programação, adaptando-se a diferentes necessidades.

**Exemplo simples:**

Lua

print("Olá, mundo!")

Use o código [com cuidado](https://asavbrm-my.sharepoint.com/faq#coding).

**Onde aprender mais sobre Lua?**

* **Site oficial:** <https://www.lua.org/portugues.html>
* **Livros:** "Programando em Lua" de Roberto Ierusalimschy é uma excelente referência.
* **Tutoriais online:** Existem diversos tutoriais gratuitos disponíveis na internet.

**Em resumo,** Lua é uma linguagem de programação poderosa, versátil e fácil de aprender. Se você está buscando uma linguagem para desenvolver jogos, sistemas embarcados, prototipar ou simplesmente aprender a programar, Lua é uma excelente opção.

**Gostaria de aprender mais sobre algum aspecto específico da linguagem Lua?**

Por exemplo, posso te explicar sobre:

* **Sintaxe:** Variáveis, operadores, estruturas de controle, funções.
* **Tabelas:** Criação, acesso, iteração.
* **Orientação a objetos:** Metaclasses, herança.
* **Módulos:** Criação e uso.
* **Bibliotecas padrão:** Coroutines, padrões, matemática.
* **Integração com outras linguagens:** C, C++.

Pesquisa sobre as linguagens orientada a objetos

Alguns exemplos de linguagens orientadas a objetos são:

Java, C++, C#, Python, Lua, PHP, Ruby, Perl, Object Pascal, Objective-C.

A programação orientada a objetos (POO) é um paradigma de programação que se baseia na ideia de que os objetos da computação são semelhantes aos objetos do mundo real. A POO é um dos paradigmas mais populares na área de programação, pois permite uma melhor visualização do que está sendo criado.

Os quatro pilares da POO são: Abstração, Encapsulamento, Herança, Polimorfismo.

A primeira linguagem de programação a implementar o conceito de objeto foi a Simula 67

**Resumo.** Este artigo tem como objetivo, fazer um breve comentário acerca da Programação Orientada a Objetos, dando uma base para aqueles que querem aprender e conhecer como ela funciona e, para aqueles que já têm conhecimento e que trabalham com programação, possam fazer comparação das aplicações da linguagem JAVA (Linguagem de programação) com as demais.

**1-INTRODUÇÃO**

A Programação Orientada a Objetos (POO) surgiu com a finalidade de facilitar a vida daqueles que trabalham com desenvolvimento de software, pois na POO o difícil não é desenvolver bem um software, mas sim desenvolver um software que satisfaça o cliente, ou seja, garantir que o que será entregue será realmente o que foi pedido. Uma das características da POO é fazer com que o programador pense as coisas de forma distintas, transformando-as assim em objeto, aplicando propriedades e métodos, que comentaremos mais adiante, reduzindo assim a complexidade no desenvolvimento e manutenção de software, aumentando a produtividade.

**2-CARATERÍSTICAS DA POO**

A POO trabalha utilizando classes, ela é quem dá a vida ao objeto, pois nela é que são criadas as funcionalidades através dos métodos, dizendo se o objeto vai andar parar rodar, etc., e suas especificidades através dos atributos, sua cor, seu nome, etc.. a classe não é o objeto, ela é utilizada para construí-lo.

O objeto é uma abstração de software que pode ser algo representado por algo real ou virtual, que é formado por um conjunto de propriedades (variáveis) e procedimentos (métodos), onde as variáveis possuem um tipo que define os possíveis valores que pode representar um número inteiro, real ou *string* e, os métodos são rotinas que, quando executadas realizam algumas tarefas como alterar o conteúdo de uma variável do objeto.A classe é a abstração do objeto. Na realidade ao programarmos um objeto, definindo suas características e funcionalidades, estamos programando uma classe.

Dentre alguns recursos aplicados na programação orientada a objeto, temos: encapsulamento, herança e polimorfismo.

Encapsulamento é a limitação imposta a atributos de uma classe, restringindo o acesso aos métodos dessa classe através de limitadores de acesso, dos quais os mais utilizados são:

Público (*public*) – a classe só é vista pelas demais classes se estiverem dentro do mesmo pacote;

Protegida (*protected*) – o acesso se restringe apenas a própria classe e suas subclasses;

Privado (*private*) – o acesso só é permitido somente pela própria classe.

O conceito de herança nos permite definir uma nova classe, com base em uma já existente. A classe criada (subclasse ou classe derivada) automaticamente herda todas as variáveis e métodos da classe já existente (superclasse). O mecanismo de herança permite que a subclasse inclua ou sobre ponha novas variáveis e métodos da superclasse. O mecanismo de herança é recursivo, permitindo criar-se uma hierarquia de classes, do nível mais alto, onde as características herdadas são comuns a todos os objetos desta classe até os níveis inferiores, onde estão as especializações das classes superiores. As subclasses herdam as características comuns, definindo suas propriedades específicas, ou seja, facilita a construção de novos objetos a partir de outros já existentes, sem a necessidade de reescrevê-lo todo.

Polimorfismo: A palavra polimorfismo utilizada na biologia para definir várias formas de membros de uma mesma espécie. Utilizando essa analogia, o mecanismo de polimorfismo utilizado na POO, nos permite tratar objetos semelhantes de uma maneira uniforme.

Para que o polimorfismo seja utilizado, é necessária a aplicação do conceito de herança, aplicados somente aos métodos da classe.

Anterior a utilização da POO, sempre que uma nova funcionalidade fosse criada ou acrescentada, a aplicação deveria ser alterada e recompilada, porém com a utilização do polimorfismo, acrescentar novos métodos a classe já existente sem a necessidade de ter que recompilar a aplicação tornou-se mais fácil.

Com base nos conceitos de objetos, classes, encapsulamento, herança e polimorfismo, que são considerados como o paradigma da POO representando uma forma inovadora de pensar e criar novos softwares, trazendo inúmeros benefícios e valores agregados no desenvolvimento deles, tais como: a reutilização de códigos que reduz o tempo e o risco no desenvolvimento, aumentando assim a confiabilidade do software, haja vista que, os códigos reutilizados já foram testados, facilita também a compreensão da linguagem com desenvolvedor na hora da manutenção de softwares complexos.

**3-CONCLUSÃO**

A linguagem de POO está em nosso dia-a-dia facilitando as nossas vidas. Muitas das vezes estamos desfrutando dos seus benefícios sem mesmo perceber, quando por exemplo: quando falamos ao celular, utilizamos o PC e, até mesmo quando estamos fazendo uma simples compra no supermercado, não temos essa percepção mas, a maioria dos programas instalados nesses equipamentos foi desenvolvido em linguagem de POO. A linguagem de POO possui ainda inúmeras ferramentas e aplicativos facilitadores que interagem ajudando os desenvolvedores e, apresenta aos iniciantes um leque de opções, dando a oportunidade de se identificarem com a linguagem de POO ou de continuar a procura de uma nova ferramenta de estudo ou de trabalho. A linguagem de POO não significa a reinvenção da roda mas, a mão na roda para qualquer desenvolvedor.

É uma aproximação à análise e desenho orientado aos objetos, centrando-se na compreensão da forma como o sistema está e será utilizado antes de ser feita a alteração pretendida.

 Orientação a Objetos (OO) é uma abordagem de programação que procura explorar nosso lado intuitivo. Os objetos da computação são análogos aos objetos existente no mundo real. ζ No enfoque de OO, os átomos do processo de computação são os objetos que trocam mensagens entre si.

**Os 4 pilares da Programação Orientada a Objetos**

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

A OOP (Object Oriented Programming -- Programação Orientada a Objetos) pode ser definida por quatro pilares principais, sendo eles herança, encapsulamento, abstração e polimorfismo.

Java

Java, como citado anteriormente, é amplamente utilizado para Programação Orientada a Objetos, pois esse é justamente seu objetivo. De maneira geral, cada linguagem tem seus pontos positivos e negativos, assim como qualquer outra coisa.

A programação orientada a objetos (POO) é um núcleo da programação Java, na qual é usada para projetar um programa usando classes e objetos. POO também podem ser caracterizados como controle de dados para acesso ao código.

Quando utilizar a Programação Orientada a Objetos  
  
Ela permite dividir o sistema em classes e objetos independentes, facilitando a compreensão e a manutenção do código. Além disso, outra vantagem do uso da POO é que a herança e a criação de hierarquias de classes permitem a reutilização eficiente de código.

Orientação a objeto é um conceito que esta relacionado com a ideia de classificar , organizar e abstrair coisas. Veja a definição formal: "O termo orientação a objetos significa organizar o mundo real como uma coleção de objetos que incorporam estrutura de dados e um conjunto de operações que manipulam estes dados.

**Nesse sentido, a POO suporta muitas linguagens de programação, entre elas:**

* C++;
* C#;
* Java;
* Object Pascal;
* VB.NET;
* Objective-C;
* Python;
* Ruby.

A Análise Orientada a Objetos (OOA) é um processo de desenvolvimento de sistemas que utiliza o conceito de objetos que interagem entre si e, através dessa interação, realizam tarefas computacionais. O ponto de partida para a OOA é criar um modelo descritivo contendo informações do projeto.

A Programação Orientada a Objetos (conhecida, também, pela sigla POO) é um paradigma de linguagem de programação e, portanto, é um dos caminhos para conseguir resolver um problema proposto. Ela auxilia a definir a forma como soluções e sistemas serão estruturados

Encapsulamento, herança e polimorfismo: as principais características da POO. As duas bases da POO são os conceitos de classe e objeto.

A programação orientada a objetos é importante por inúmeras razões, entre elas: Reutilização de código: A POO promove a reutilização de código por meio do conceito de herança, possibilitando que classes derivadas herdam características e comportamentos de classes base.

A Programação Orientada a Objetos (POO) diz respeito a um padrão de desenvolvimento que é seguido por muitas linguagens, como C# e Java. A seguir, iremos entender as diferenças entre a POO e a Programação Estruturada, que era muito utilizada há alguns anos, principalmente com a linguagem C.

No final dos anos 80, Bjarne Stroustrup estendeu a linguagem C com alguns conceitos de objetos e criou C++. Em termos de orientação a objetos, pode-se dizer que C++ está no extremo oposto de Smalltalk. realmente OO, influenciada por Smalltalk.

Em primeiro lugar, a POO promove uma maior organização e modularização do código, tornando-o mais fácil de entender, manter e reutilizar. Além disso, a POO permite a criação de programas mais escaláveis e flexíveis, pois os objetos podem ser facilmente modificados ou estendidos sem afetar o resto do sistema.

O Conceito de Abstração em POO  
  
Em essência, a abstração permite aos desenvolvedores criar modelos simplificados de entidades complexas do mundo real, focando apenas nos aspectos que são importantes para a aplicação sendo desenvolvida.

**Pilares da POO**

* Figura 1: **Pilares da POO**.
* Abstração.
* A abstração é o processo de ocultar os detalhes complexos e mostrar apenas as características essenciais de um objeto ou sistema. ...
* Encapsulamento.
* O encapsulamento é a habilidade de esconder as características intrínsecas de um dado objeto de outros objetos. ...
* Herança.

As variáveis de uma classe só podem ser alteradas por métodos definidos nos seus objetos. O polimorfismo se caracteriza quando, para mensagens distintas, objetos diferentes responderem ou agirem de forma idêntica.

Outras linguagens como C, Lisp e Erlang nem sequer se importam em ser ou parecer orientadas a objetos.

JavaScript, Python e PHP estão entre as linguagens de programação mais usadas.

Malbolge

A maior característica de Malbolge é que ela foi especialmente criada para ser a pior linguagem de programação possível (a mais difícil e mais esotérica).

A POO promove a reutilização de código, modularidade, legibilidade e manutenibilidade do código, sendo suportada por linguagens como: Java, C#, C++, Python, PHP, Ruby e Object Pascal.

A programação orientada a objetos é um modelo de programação onde diversas classes possuem características que definem um objeto na vida real. Cada classe determina o comportamento do objeto definido por métodos e seus estados possíveis definidos por atributos.

O principal paradigma da linguagem Python é a orientação a objetos, o que facilita muito a compreensão de alguns conceitos, torna a programação mais simples de ser entendida, facilita o trabalho em equipe, além de proporcionar um melhor controle e maior estabilidade na produção de grandes projetos.

Desvantagens da Programação Orientada a Objetos  
  
Apesar dos benefícios, a POO pode ser mais complexa e ter uma curva de aprendizado mais íngreme. A criação e manipulação de objetos podem adicionar uma sobrecarga ao desempenho do programa.

Com a POO, é possível desenvolver programas eficientes, modularizados e de fácil manutenção, partindo das classes, objetos, propriedades e métodos, resultando assim em um código muito mais organizado e reutilizável

A primeira linguagem de programação a implementar sistematicamente os conceitos de OOP foi a linguagem SIMULA-68; em seguida surgiu a linguagem Smalltalk -criada pela Xerox -, que pode ser considerada a linguagem que popularizou e incentivou o emprego da OOP.

Conclusão. Neste guia completo para iniciantes, exploramos os conceitos básicos da Programação Orientada a Objetos e os quatro pilares que sustentam essa abordagem. Através da abstração, encapsulação, herança e polimorfismo, podemos criar sistemas de software mais eficientes, flexíveis e modulares.

Neste artigo, você irá conhecer o conceito de Programação Orientada a Objetos (POO), o que são classes e objetos, além de vantagens e pilares da POO.

Imagine que você precisa desenvolver um jogo de computador com dezenas de personagens com características diferentes, dentre eles zumbis, vampiros e fantasmas. Você escreveria o código de cada um deles a partir do zero ou buscaria formas de reaproveitar um único código para os demais?

Se você escolheu a segunda opção, conscientemente ou não, optou pela **Programação Orientada a Objetos**. Esse é o paradigma de programação mais popular atualmente, pois além de ser utilizado em diversas linguagens de programação, ele te permite escrever códigos de forma mais eficiente, principalmente em projetos de software de grande porte, como o do exemplo acima.

## O que é Programação Orientada a Objetos?

**Programação Orientada a Objetos (POO)** é um **paradigma de programação** que se baseia na utilização de **objetos**para representar elementos do mundo real e seus comportamentos.

Esses objetos são modelados como entidades autônomas, pois possuem atributos e comportamentos independentes. Ou seja, cada objeto é uma entidade única, com dados (atributos) e ações (métodos) específicos para ele.

A Programação Orientada a Objetos é uma abordagem mais eficiente e intuitiva para a modelagem de sistemas complexos. Antes dela, a programação era feita principalmente através de procedimentos, o que tornava o desenvolvimento de sistemas grandes e complexos uma tarefa difícil e confusa.

Portanto, a POO surgiu para solucionar esses problemas, oferecendo uma maneira de representar objetos do mundo real como objetos em código, com atributos e comportamentos específicos. Além disso, a POO oferece recursos para reutilização de código, encapsulamento, herança e polimorfismo, tornando a programação mais organizada e eficiente.

Esta é uma abordagem flexível e escalável que tem se mostrado útil para a modelagem de sistemas em uma ampla variedade de aplicações, desde pequenos projetos até grandes sistemas corporativos.

## O que são classes e objetos?

No universo da POO, as noções de classes e objetos desempenham um papel fundamental. Elas proporcionam uma estrutura organizada para modelar e representar entidades do mundo real no âmbito do desenvolvimento de software. Vamos explorar esses conceitos essenciais que formam a base da programação orientada a objetos.

### Classe

**Classe**é uma representação abstrata de um objeto ou conceito do mundo real, com atributos (variáveis) e métodos (comportamentos) definidos. Esses métodos também recebem o nome de **funções internas da classe**. Em outras palavras, uma classe é o tipo de dado a ser criado, sendo uma espécie de “molde” que o programador utiliza para criar objetos similares.

### Objeto

**Objeto**é o elemento individual criado dentro de uma classe. Também conhecido como **instância,**possui atributos e métodos. Ou seja, é uma entidade concreta baseada no molde da classe. O processo de criação de um objeto chama-se **instanciação**.

Retomemos o exemplo do programa de jogo de computador do início do artigo. Nele, você precisa desenhar vários tipos de inimigos na tela, como zumbis, vampiros e fantasmas. Então, em vez de escrever o código para desenhar cada inimigo a partir do zero, você pode criar uma **classe** “Inimigo” com as propriedades comuns a todos os inimigos (como posição na tela, vida, dano), e depois criar **subclasses específicas** para cada tipo de inimigo (como “Zumbi”, “Vampiro”, “Fantasma”), os quais herdam as propriedades da classe “Inimigo” e adicionam propriedades exclusivas para cada tipo (como aparência diferente).

Dessa forma, você pode reutilizar o código da classe “Inimigo” para todos os tipos de inimigos, economizando tempo e tornando o código mais fácil de manter.

**Quais são as vantagens da POO?**

A Programação Orientada a Objetos tem várias vantagens que a tornam importante para programadores aprender, incluindo:

* **Reutilização de código:**permite a reutilização do código em vários lugares, o que economiza tempo e aumenta a eficiência;
* **Facilidade de manutenção:**ajuda a organizar o código em unidades menores e mais fáceis de entender, tornando-o mais fácil de manter e corrigir;
* **Melhor separação de responsabilidades:** permite a distribuição das tarefas de maneira mais clara entre diferentes objetos;
* **Maior capacidade de escalabilidade:** ajuda a tornar o código escalável, o que é importante em projetos de software de grande porte;
* **Maior facilidade de colaboração:**permite que vários programadores trabalhem  juntos em projetos de software de maneira mais eficiente.

**Os quatro pilares da POO**

Para uma linguagem de programação poder atender ao paradigma da Programação Orientada a Objetos, ela precisa seguir quatro pilares. Vejamos todos eles e como se aplicam ao nosso exemplo do jogo de computador.

**1. Abstração:** é a capacidade de representar somente as características essenciais de um objeto.

* Exemplo: cria-se uma classe abstrata “Personagem” para representar as características comuns a todos os personagens do jogo, como “nome”, “saúde”, “ataque” e “defesa”. Esta classe seria a base para as subclasses específicas de cada personagem.

**2. Encapsulamento:** os detalhes de implementação dos objetos podem ser escondidos.

* Exemplo: protege-se os atributos e comportamentos de cada personagem, restringindo o acesso a partir de outras classes ou objetos. No caso, pode-se acessar a saúde de um personagem somente por métodos específicos, evitando a manipulação inadequada.

**3. Herança:**é arelação entre classes na qual uma classe herda as características de outra.

* Exemplo: cria-se classes específicas para cada tipo de personagem, como “Zumbi”, “Vampiro” e “Fantasma”, que herdem as características da classe “Personagem” e adicionam comportamentos e atributos específicos. Assim, um zumbi pode ter uma velocidade de movimento mais baixa do que um vampiro. Saiba mais no tutorial [Herança em Classes: Como Usar em Python](https://hub.asimov.academy/tutorial/heranca-em-classes-como-usar-em-python/?utm_custom=blog).

**4. Polimorfismo:** é acapacidade de objetos de diferentes classes serem tratados de forma semelhante.

* Exemplo: utiliza-se o polimorfismo para que diferentes personagens respondam de maneiras diferentes a uma mesma mensagem, como “ataque”. Dessa forma, um zumbi pode realizar ataques físicos, enquanto um fantasma pode realizar ataques à distância com sua habilidade de passar através de obstáculos.

**Quais são as linguagens de Programação Orientada a Objetos?**

Muitas são as linguagens de programação que permitem a Programação Orientada a Objetos. As principais são:

* [*Python*](https://www.python.org/?utm_custom=blog)
* *Java*
* *JavaScript*
* *C++*
* *Ruby*
* *Swift*
* *C#*
* *PHP*
* *Kotlin*
* *R*
* *Objective-C*

Portanto, os conceitos vistos aqui se aplicam a diferentes tipos de linguagens.

**Como aprender Programação Orientada a Objetos?**

Para aprender sobre Programação Orientada a Objetos, é importante estudar [lógica de programação](https://asimov.academy/logica-de-programacao-o-que-e-como-funciona/?utm_custom=blog) e**paradigmas de programação** antes de entrar na teoria e nos conceitos fundamentais da POO.

Para praticar, escolha uma linguagem que permita a POO e dedique-se, pois estudar esse tema é muito valioso para os programadores. Ao aprender sobre diferentes abordagens, eles podem escolher a melhor maneira de solucionar problemas de programação em situações diferentes e escrever códigos mais eficientes.

Dart é uma linguagem de programação multiparadigma e multiplataforma, criada pela Google em 2011. A linguagem foi concebida para substituir o JavaScript como a linguagem principal embutida nos navegadores.

|  |  |
| --- | --- |
|  | **Características** |
| Tipagem | Estática, o que significa que o tipo de dado de uma variável deve ser declarado explicitamente |
| Paradigma | Orientado a objetos |
| Execução | Pode ser compilada em ahead-of-time (AOT) ou just-in-time (JIT) |
| Sintaxe | C-like, o que facilita o aprendizado para quem já programa em Java, C#, PHP ou Javascript |
| Ambiente de execução | Pode ser executada em uma máquina virtual (DartVM), ou transpilada para JavaScript através da ferramenta dart2js |

O Dart é usado para desenvolver aplicações web, móveis e de desktop. A linguagem é flexível e pode ser executada em ambientes nativos ou web.

O Dart passou a ser uma das linguagens preferidas a partir de 2015, quando a Google lançou o Flutter, que é fundamentado no Dart.

Para começar a estudar Dart, é recomendado estar sempre atento à documentação oficial, pois a linguagem está em constante amadurecimento.

O que é Dart? Dart é uma linguagem de programação desenvolvida pela Google, semelhante a Java, C++, e JavaScript. Ela é "levemente tipada" e pode ser tanto interpretada quanto compilada.

Kotlin é uma linguagem de programação multiplataforma, orientada a objetos e funcional, concisa e estaticamente tipada, desenvolvida em 2011 pela empresa tcheca JetBrains, que compila para a Máquina virtual Java e também traduzida para a linguagem JavaScript e compilada para código nativo

Kotlin é uma linguagem oficialmente suportada para o desenvolvimento de aplicativos Android. Desde maio de 2017, o Google anunciou o suporte total a Kotlin para o desenvolvimento Android, e muitos desenvolvedores têm adotado a linguagem devido à sua concisão e expressividade em comparação com Java.

Kotlin é uma linguagem de tipagem estática, ou seja, as variáveis e funções precisam ser declaradas com um tipo específico. Já Java é uma linguagem de tipagem dinâmica, em que as variáveis e funções não precisam ter um tipo definido.