RESUMO DE PROGRAMAÇÃO ORIENTADA A OBJETOS

O QUE É PROGRAMAÇÃO ORIENTADA A OBJETOS?

Programação Orientada a Objetos (POO) é um paradigma de programação que se baseia no conceito de objetos, que são entidades que possuem propriedades e comportamentos. Em POO, um programa é organizado em torno de objetos que interagem entre si para realizar tarefas. Cada objeto tem um estado interno, que é definido por suas propriedades, e um conjunto de operações que podem ser realizadas sobre ele, que são definidas por seus métodos.

POO permite a criação de programas modulares, flexíveis e reutilizáveis. Com POO, é possível abstrair a complexidade de um sistema, dividindo-o em objetos mais simples e interconectados, o que facilita o desenvolvimento e a manutenção de programas. Além disso, POO suporta conceitos como encapsulamento, herança e polimorfismo, que permitem que os programadores escrevam código mais limpo, seguro e extensível. POO é amplamente utilizado em linguagens de programação modernas, como Java, Python e C++.

PILARES DA PROGRAMAÇÃO ORIENTADA A OBJETOS (POO):

1. **Abstração:** refere-se à capacidade de abstrair ou extrair as características essenciais de um objeto do mundo real, a fim de representá-lo em um programa de computador. A abstração permite que os programadores ignorem os detalhes irrelevantes e se concentrem apenas nas características mais importantes do objeto.
2. **Encapsulamento:** refere-se à capacidade de ocultar a complexidade interna de um objeto e fornecer uma interface externa simples e consistente para o usuário. O encapsulamento ajuda a garantir que o objeto seja usado corretamente, protegendo suas propriedades e métodos internos de acesso não autorizado.
3. **Herança:** refere-se à capacidade de criar novas classes com base em classes existentes, herdando suas características e adicionando novas. A herança permite que os programadores reutilizem código existente, economizando tempo e esforço no desenvolvimento de novos programas.
4. **Polimorfismo:** refere-se à capacidade de usar um mesmo nome para representar diferentes comportamentos. O polimorfismo permite que os programadores escrevam código genérico que pode ser usado com diferentes tipos de objetos, aumentando a flexibilidade e a reutilização de código.

CLASSE

Uma classe na Programação Orientada a Objetos (POO) é uma estrutura que define um tipo de objeto. Ela é uma "receita" ou um "modelo" para criar objetos que possuem propriedades e comportamentos específicos.

Na POO, uma classe é uma abstração que define um conjunto de propriedades e métodos que são comuns a um grupo de objetos. Por exemplo, uma classe "Carro" pode ter propriedades como "modelo", "ano", "cor" e "marca", bem como métodos como "acelerar", "frear" e "ligar". Essas propriedades e métodos podem ser usados para criar vários objetos de carro que compartilham as mesmas características e comportamentos.

RESUMO DE PROGRAMAÇÃO ORIENTADA A OBJETOS

Em resumo, uma classe é a definição de um tipo de objeto, que contém seus atributos e métodos, e a partir dela é possível criar vários objetos (instâncias) que compartilham as mesmas características.

OBJETOS

Na Programação Orientada a Objetos (POO), um objeto é uma instância de uma classe. Em outras palavras, é um "exemplar" ou "cópia" de uma classe que possui seus próprios valores para as propriedades definidas na classe.

Em resumo, um objeto é uma instância de uma classe, que possui seus próprios valores para as propriedades definidas na classe e compartilha os mesmos métodos. Ele contém dados e comportamentos associados a esses dados, permitindo que o programador trabalhe com dados de forma mais intuitiva e natural.

ABSTRAÇÃO

Abstração na Programação Orientada a Objetos (POO) é o processo de identificar as características essenciais de um objeto ou sistema e representá-las de forma simplificada e conceitual, ignorando detalhes irrelevantes ou complexos. Em outras palavras, é o processo de criar modelos simples que representam objetos ou sistemas complexos.

A abstração permite que o programador foque nos aspectos relevantes de um objeto ou sistema e oculte os detalhes complexos, tornando o desenvolvimento de software mais fácil, flexível e adaptável. Ela ajuda a criar uma separação clara entre a interface pública de um objeto e sua implementação interna, permitindo que as mudanças sejam feitas na implementação sem afetar os usuários da classe.

Na POO, a abstração é geralmente alcançada por meio da criação de classes abstratas ou interfaces. Classes abstratas são classes que não podem ser instanciadas, mas podem ser usadas como base para outras classes que implementam seus métodos. Interfaces são semelhantes às classes abstratas, mas definem apenas a assinatura dos métodos, deixando a implementação para as classes que a implementam.

Em resumo, a abstração na POO é o processo de identificar as características essenciais de um objeto ou sistema e representá-las de forma simplificada e conceitual, permitindo que o programador se concentre nos aspectos relevantes e oculte os detalhes complexos. Ela é geralmente alcançada por meio da criação de classes abstratas ou interfaces.

ATRIBUTOS

Na Programação Orientada a Objetos (POO), um atributo é uma variável que armazena dados em um objeto. É uma característica ou propriedade que define o estado de um objeto.

Os atributos também são conhecidos como "propriedades" ou "variáveis de instância". Eles são definidos dentro de uma classe e podem ser acessados pelos objetos criados a partir dessa classe. Cada objeto tem seus próprios valores de atributos, mesmo que sejam criados a partir da mesma classe.

Por exemplo, em uma classe "Carro", podemos definir atributos como "modelo", "ano", "cor" e "marca". Cada objeto "Carro" criado a partir dessa classe terá seus próprios valores para esses atributos.

Os atributos podem ter diferentes tipos de dados, como inteiros, floats, strings, booleanos, etc. Podem ser definidos como públicos (acessíveis de fora da classe), privados (acessíveis apenas de dentro da classe) ou protegidos (acessíveis apenas pela classe e suas subclasses).

Em resumo, um atributo é uma variável que armazena dados em um objeto na POO. Eles são definidos dentro de uma classe e podem ser acessados pelos objetos criados a partir dessa classe. Cada objeto tem seus próprios valores de atributos.

MÉTODOS

Na Programação Orientada a Objetos (POO), um método é uma função que pertence a uma classe ou objeto e é usado para realizar operações específicas. Em outras palavras, um método é um bloco de código que contém uma série de instruções que podem ser chamadas pelos objetos para executar uma tarefa específica.

Os métodos são responsáveis por manipular os dados do objeto e executar as operações necessárias para implementar a funcionalidade da classe. Eles são definidos dentro da classe e podem ser chamados pelos objetos criados a partir dessa classe.

Os métodos podem ter diferentes tipos de visibilidade, como público, privado ou protegido. Métodos públicos são acessíveis de fora da classe e podem ser chamados pelos objetos. Métodos privados só podem ser chamados de dentro da classe. Métodos protegidos são semelhantes aos privados, mas podem ser acessados pelas subclasses da classe.

Os métodos podem receber parâmetros como entradas e retornar valores como saídas. Eles podem ser sobrecarregados, o que significa que podem ter o mesmo nome, mas parâmetros diferentes e/ou tipos de retorno diferentes.

Em resumo, um método é uma função que pertence a uma classe ou objeto e é usado para realizar operações específicas na POO. Eles são responsáveis por manipular os dados do objeto e executar as operações necessárias para implementar a funcionalidade da classe. Os métodos podem ter diferentes tipos de visibilidade, receber parâmetros e retornar valores.

MÉTODO CONSTRUTOR

Métodos construtores são métodos especiais nas classes da Programação Orientada a Objetos (POO) que são usados para inicializar os objetos quando são criados. Eles são responsáveis por atribuir valores iniciais aos atributos da classe e executar outras operações de inicialização necessárias.

O método construtor é chamado automaticamente quando um objeto é criado a partir da classe correspondente. Ele pode ser usado para inicializar os atributos da classe com valores padrão ou com valores fornecidos pelo usuário.

Na maioria das linguagens de programação orientadas a objetos, o método construtor tem o mesmo nome da classe e não possui tipo de retorno. No entanto, alguns idiomas podem ter convenções diferentes, como o uso de palavras-chave especiais para identificar o construtor.

Além disso, uma classe pode ter vários métodos construtores com diferentes parâmetros. Isso é conhecido como sobrecarga de construtores. Os objetos podem ser criados com diferentes conjuntos de parâmetros, dependendo do construtor chamado.

Em resumo, os métodos construtores são usados para inicializar os objetos quando são criados na POO. Eles são responsáveis por atribuir valores iniciais aos atributos da classe e executar outras operações de inicialização necessárias. Na maioria das linguagens de programação orientadas a objetos, o método construtor tem o mesmo nome da classe e não possui tipo de retorno.

ENCAPSULAMENTO

Na Programação Orientada a Objetos (POO), encapsulamento é o conceito de esconder os detalhes de implementação de uma classe e expor apenas uma interface pública para interagir com ela.

Em outras palavras, o encapsulamento é uma técnica que permite que a implementação interna de um objeto seja oculta do mundo exterior e somente os métodos públicos que fazem parte da interface da classe são expostos para uso externo. Isso significa que outras partes do código não têm acesso direto aos dados e funções internas da classe e só podem interagir com ela através dos métodos públicos que foram explicitamente expostos.

O encapsulamento é importante porque ajuda a proteger o estado interno da classe de ser alterado indevidamente ou corrompido por outras partes do código. Ele também permite que os desenvolvedores alterem a implementação interna de uma classe sem afetar o código que a utiliza, desde que a interface pública permaneça a mesma.

CÓDIGOS

Aqui está um exemplo simples de uma classe em Python que representa um objeto "Pessoa" com alguns atributos e métodos:

EM PYTHON

```
class Pessoa:
    def __init__(self, nome, idade):
        self.nome = nome
        self.idade = idade

    def saudacao(self):
        print("Olá, meu nome é", self.nome, "e eu tenho", self.idade,
"anos.")

    def aniversario(self):
        self.idade += 1
        print("Feliz aniversário! Agora eu tenho", self.idade, "anos.")

# Criando uma instância da classe Pessoa
pessoa1 = Pessoa("João", 25)

# Chamando o método saudacao()
pessoa1.saudacao() # Saída: "Olá, meu nome é João e eu tenho 25 anos."

# Chamando o método aniversario()
pessoa1.aniversario() # Saída: "Feliz aniversário! Agora eu tenho 26
anos."
```

Neste exemplo, a classe `Pessoa` tem dois atributos, `nome` e `idade`, e dois métodos, `saudacao()` e `aniversario()`. O método `__init__()` é um método especial chamado de construtor que é executado automaticamente quando uma nova instância da classe é criada. Os métodos `saudacao()` e `aniversario()` são métodos de instância, ou seja, eles operam no objeto específico que foi criado a partir da classe. Note que os métodos têm acesso aos atributos da instância através do parâmetro `self`.

Finalmente, criamos uma instância da classe `Pessoa` chamada `pessoa1` e chamamos os métodos `saudacao()` e `aniversario()` para interagir com o objeto.

RESUMO DE PROGRAMAÇÃO ORIENTADA A OBJETOS

EM JAVA

```java
public class Pessoa {
    private String nome;
    private int idade;

    public Pessoa(String nome, int idade) {
        this.nome = nome;
        this.idade = idade;
    }

    public void saudacao() {
        System.out.println("Olá, meu nome é " + nome + " e eu tenho " +
idade + " anos.");
    }

    public void aniversario() {
        idade++;
        System.out.println("Feliz aniversário! Agora eu tenho " + idade +
" anos.");
    }

    public static void main(String[] args) {
        Pessoa pessoa1 = new Pessoa("João", 25);
        pessoa1.saudacao(); // Saída: "Olá, meu nome é João e eu tenho 25
anos."
        pessoa1.aniversario(); // Saída: "Feliz aniversário! Agora eu
tenho 26 anos."
    }
}
```

Neste exemplo, a classe `Pessoa` tem dois atributos, `nome` e `idade`, e dois métodos, `saudacao()` e `aniversario()`. O modificador de acesso `private` é usado para tornar os atributos acessíveis apenas dentro da classe `Pessoa`. O método construtor `Pessoa(String nome, int idade)` é executado automaticamente quando uma nova instância da classe é criada. Os métodos `saudacao()` e `aniversario()` são métodos de instância, ou seja, eles operam no objeto específico que foi criado a partir da classe. Note que os métodos têm acesso aos atributos da instância usando o operador `this`.

Finalmente, criamos uma instância da classe `Pessoa` chamada `pessoa1` e chamamos os métodos `saudacao()` e `aniversario()` para interagir com o objeto. O método `main()` é um método especial em Java que é executado automaticamente quando o programa é iniciado. Ele cria uma nova instância da classe `Pessoa` e chama seus métodos.

CÓDIGO ENCAPSULAMENTO

EM PYTHON

```python
class ContaBancaria:
    def __init__(self, saldo_inicial):
        self.__saldo = saldo_inicial

    def deposito(self, valor):
        self.__saldo += valor

    def saque(self, valor):
        if valor <= self.__saldo:
            self.__saldo -= valor
        else:
            print("Saldo insuficiente.")

    def get_saldo(self):
        return self.__saldo

# Criando uma instância da classe ContaBancaria
conta1 = ContaBancaria(1000)

# Tentando acessar o atributo __saldo diretamente (erro!)
# print(conta1.__saldo) # Erro: 'ContaBancaria' object has no attribute
'__saldo'

# Chamando o método deposito() para adicionar dinheiro à conta
conta1.deposito(500)

# Chamando o método saque() para retirar dinheiro da conta
conta1.saque(200)

# Chamando o método get_saldo() para obter o saldo atual
print("O saldo atual é:", conta1.get_saldo()) # Saída: "O saldo atual é:
1300"
```

EM JAVA

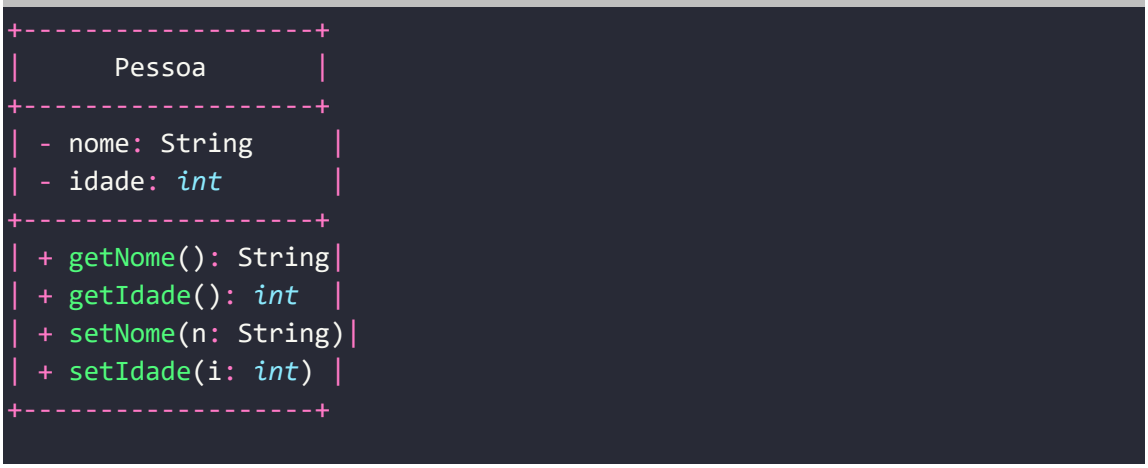
```java
public class ContaBancaria {
    private double saldo;

    public ContaBancaria(double saldo_inicial) {
        saldo = saldo_inicial;
    }

    public void deposito(double valor) {
        saldo += valor;
    }

    public void saque(double valor) {
        if (valor <= saldo) {
            saldo -= valor;
        } else {
            System.out.println("Saldo insuficiente.");
        }
    }

    public double getSaldo() {
        return saldo;
    }

    public static void main(String[] args) {
        ContaBancaria conta1 = new ContaBancaria(1000);
        conta1.deposito(500);
        conta1.saque(200);
        System.out.println("O saldo atual é: " + conta1.getSaldo()); //
Saída: "O saldo atual é: 1300.0"
    }
}
```

Neste exemplo, a classe `ContaBancaria` tem um atributo privado `saldo` que é encapsulado usando o modificador de acesso `private`. Isso significa que o atributo não pode ser acessado diretamente fora da classe - qualquer acesso deve ser feito por meio de um método público. A classe também tem dois métodos públicos, `deposito()` e `saque()`, que atualizam o saldo da conta, e um método público `getSaldo()` que retorna o saldo atual.

Ao criar uma instância da classe `ContaBancaria`, o saldo inicial é passado para o construtor `ContaBancaria(double saldo_inicial)`. Tentar acessar o atributo `saldo` diretamente fora da classe resultará em um erro devido ao modificador de acesso `private`. Em vez disso, usamos os métodos `deposito()`, `saque()` e `getSaldo()` para interagir com o objeto. Isso protege o atributo `saldo` de ser alterado indevidamente ou corrompido por outras partes do código.

RESUMO DE PROGRAMAÇÃO ORIENTADA A OBJETOS

DIAGRAMA DE UMA CLASSE PESSOA



Neste exemplo, temos uma classe chamada "Pessoa" com dois atributos privados: "nome" e "idade". A classe também possui quatro métodos públicos: "getNome()", "setNome(nome)", "getIdade()" e "setIdade(idade)". O método "getNome()" retorna o valor atual do atributo "nome", o método "setNome(nome)" define o valor do atributo "nome", o método "getIdade()" retorna o valor atual do atributo "idade" e o método "setIdade(idade)" define o valor do atributo "idade". O sinal "+" antes dos métodos indica que eles são públicos.