PROGRAMAÇÃO ORIENTADA A OBJETOS II

Profª Letícia Pieper Jandt

EMENTA

Conceitos e evolução da programação orientada a objetos. Limitações e diferenças entre o paradigma da programação estruturada em relação à orientação a objetos. Abstração e modelo conceitual. Conceito e modelos de objetos. Classes, atributos, métodos, mensagens/ações. Construtores e destrutores. Polimorfismo.

Herança – simples e múltipla e suas consequências. Encapsulamento e ocultamento de dados. Conceitos e técnicas de programação. Implementação de algoritmos orientado a objetos utilizando linguagens de programação. Múltiplas linhas de execução. Java x Applets. Exceções e depuração. Aplicação e uso das estruturas fundamentais da orientação a objetos. Aplicações em ambiente WEB. Criação de projeto.

- No semestre:
  - NODE API
  - Conceito e Construção de API

VAMOS RELEMBRAR

ABSTRAÇÃO

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. Por exemplo, a classe Animal pode ter os atributos nome, espécie e idade, e os métodos comer e dormir.

VAMOS RELEMBRAR

ABSTRAÇÃO

Podemos destacar três principais pontos da abstração na POO:

Identidade: Cada objeto deve ter uma identidade única dentro do sistema para que não haja conflito.

Propriedades: Os objetos podem ter propriedades que os definem.

Métodos E funções: Os objetos podem ter métodos e funções que representam as ações que eles podem realizar.

VAMOS RELEMBRAR

ENCAPSULAMENTO

O encapsulamento na POO é a técnica de esconder as propriedades de um objeto, criando uma caixa preta que protege os dados internos. Ele é implementado por meio de propriedades privadas, que só podem ser acessadas ou modificadas por meio de métodos especiais chamados getters e setters. Essa técnica evita o acesso direto às propriedades do objeto, aumentando a segurança da aplicação.

VAMOS RELEMBRAR

ENCAPSULAMENTO

O encapsulamento pode ser comparado com o que acontece no mundo real. Por exemplo, quando ligamos a televisão, não precisamos saber como ela funciona internamente. Podemos dizer que os métodos que ligam a televisão estão encapsulados.

VAMOS RELEMBRAR

ENCAPSULAMENTO

Os modificadores de acesso mais comuns são:

public: os dados e funções que possuem o modificador *public* podem ser acessados por qualquer objeto.

protected: os dados e funções que possuem o modificador *protected* podem ser acessados por classes que são subclasses da classe que os define.

private: os dados e funções que possuem o modificador *private* só podem ser acessados pela classe que os define.

VAMOS RELEMBRAR

HERANÇA

A herança é um conceito da programação orientada a objetos que permite criar novas classes a partir de outras já existentes, aproveitando seus atributos e métodos. A herança facilita o reuso de código, a organização e a manutenção do software.

VAMOS RELEMBRAR

HERANÇA

Um exemplo é a classe Pessoa, que pode ter os atributos nome e matrícula, e ser a classe pai de outras classes, como Aluno, Professor e Funcionário. Essas classes filhas podem herdar os atributos de Pessoa e adicionar outros específicos, como curso, salário e cargo.

VAMOS RELEMBRAR

POLIMORFISMO

Definimos Polimorfismo como um princípio a partir do qual as classes derivadas de uma única classe base são capazes de invocar os métodos que, embora apresentem a mesma assinatura, comportam-se de maneira diferente para cada uma das classes derivadas.

VAMOS RELEMBRAR

POLIMORFISMO

O Polimorfismo é um mecanismo por meio do qual selecionamos as funcionalidades utilizadas de forma dinâmica por um programa no decorrer de sua execução.

VAMOS RELEMBRAR

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, mais que o cálculo deles em uma venda é calculado de formas diferentes.

VAMOS RELEMBRAR

Com o Polimorfismo, os mesmos atributos e objetos podem ser utilizados em objetos distintos, porém, com implementações lógicas diferentes.

VAMOS RELEMBRAR

Por 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.

VAMOS RELEMBRAR

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.

VAMOS RELEMBRAR

Assim podemos ter na classe base o método CalcularVendas:

```
public decimal CalcularVendas()  {
    decimal valorUnitario = decimal.MinValue;
    decimal produtosVendidos = decimal.MinValue;
    return valorUnitario * produtosVendidos;
 }
```

VAMOS RELEMBRAR

Na classe Vendedor temos o mesmo método, mais com a codificação diferente:

```
public decimal CalcularVendas(){
    decimal valorUnitario = 50;
    decimal produtosVendidos = 1500;
    return valorUnitario * produtosVendidos;
 }
```

VAMOS RELEMBRAR

O mesmo ocorre na classe Diretor:

```
public decimal CalcularVendas(){
    decimal valorUnitario = 150;
    decimal produtosVendidos = 3800;
    decimal taxaAdicional = 100;
    return taxaAdicional + (valorUnitario * produtosVendidos);
 }
```

VAMOS RELEMBRAR

tudo depende de que classe o método é chamado.

Projeção de Sistemas

Faça a projeção de um sistema de vendas onde o co sistema é:

- Pessoa
  - Cliente
  - Vendedor
  - Gerente
- Produto
  - matéria prima
  - produto acabado
  - estoque
- Vendas
  - Pedidos
  - Venda finalizada

Classes com seus objetos e funções
Abstração
Encapsulamento
Herança

Projeção de Sistemas

Faça a projeção de um sistema de controle de turmas onde o co sistema é:

- Pessoa
  - Aluno
  - Professor
  - Diretor
  - Coordenador
- Semestre
- Matérias
- Relação Aluno x semestre
- Turmas
- Notas

Classes com seus objetos e funções
Abstração
Encapsulamento
Herança

Projeção de Sistemas

Faça a projeção de um sistema de locação onde o co sistema é:

- Pessoa
  - Cliente
  - Locadora
- Carro
  - modelo
  - ano
- Locação
  - Período x aluguel

Classes com seus objetos e funções
Abstração
Encapsulamento
Herança

Projeção de Sistemas

- Enviar via email

  leticia.pieper1@gmail.como

https://www.dio.me/articles/pilares-de-poo-em-java

https://www.devmedia.com.br/conceitos-e-exemplos-polimorfismo-programacao-orientada-a-objetos/18701