**Teste**

**tes**

**CONCEITOS DE PROGRAMACAO OO**

**- DIFERENÇA ENTRE O PROGRAMAÇÃO ESTRUTURADA E A OO:**

**ESTRUTURADA:**

- Segue uma formula básica de: sequência -> decisão -> repetição;

- Procedimentos (ou funções) que são aplicados globalmente (informações armazenas em variáveis globais);

- Paradigma estruturado costuma permitir mais liberdades com o hardware, o que acaba auxiliando na questão desempenho;

**OO:**

- Dados e funções relacionadas são encapsuladas em um mesmo elemento denominado OBJETO;

- Temos métodos (semelhante a funções em C) que são aplicados aos dados de cada objeto;

**Vantagens:**

- Maior reaproveitamento;

- Robustez;

- Maior facilidade de manutenção;

- Maior gerenciamento;

**Conceitos:**

* **OBJETO**

- Trata-se de uma ocorrência específica de uma classe; é uma “instância de classe”;

- Os objetos representam entidades do mundo real, como aviões, pessoas, contas correntes, etc. Um objeto possui características próprias (atributos) e executa determinadas ações (métodos), sendo esses atributos e métodos provenientes da classe que origina o objeto.

**-** Formaliza o modo pelo qual compreendemos algo do domínio do problema;

* **CLASSE**

**-** Conjunto de objetos que são com os mesmos atributos e as mesmas operações. Uma classe representa uma idéia ou conceito e classifica objetos que tenham propriedades similares. As classes representam os blocos de construção mais importantes dos sistemas orientados a objetos;

**-** As classes são constituídas por atributos (dados que o objeto armazenara. Ao criar o objeto, se criara o espaço de memoria para todos os atributos dele) e por métodos (funções) que são operações executadas por objetos e tem acesso direto a todos os atributos do objeto;

* **ABSTRAÇÃO**

- Significa selecionar aspectos específicos de um problema a ser analisado, deixando de lado outros aspectos**.** *Pelo princípio da abstração, nós isolamos os objetos que queremos representar do ambiente complexo em que se situam, e nesses objetos representamos somente as características que são relevantes para o problema em questão”.*

* **HERANÇA**

- É a capacidade de um novo objeto tomar atributos e operações de um objeto existente. Usamos a herança, por exemplo, para evitar repetição ao definirmos classes com características em comum e que são relacionadas entre si.

* **ENCAPSULAMENTO**

- É a combinação de atributos e operações dentro de uma classe, deixando visível apenas o que é necessário para a comunicação entre dois objetos, como por exemplo alguns detalhes da implementação ou ainda a lógica de um método. O encapsulamento permite ocultar a complexidade do código. Não é necessário entender como a operação funciona para poder utilizá-la.

* **POLIMORFISMO**

- Significa muitas formas, na orientação a objetos você pode enviar uma mesma mensagem para diferentes objetos e fazê-los responder da maneira correta. "Polimorfismo é o princípio pelo qual duas ou mais classes derivadas de uma mesma superclasse podem invocar métodos que têm a mesma identificação (assinatura), mas comportamentos distintos, especializados para cada classe derivada, usando para tanto uma referência a um objeto do tipo da superclasse"

**PARADIGMAS DE PROGRAMAÇÃO**

-São formas de classificar as **linguagens de programação** baseado em suas **características e funcionalidades**.

-Linguagens podem ser classificadas em múltiplos paradigmas;

* 1. [imperativa](https://en.wikipedia.org/wiki/Imperative_programming) - define a sequência das operações
     1. Chamada também de programação algorítmica
     2. Descreve a computação em detalhes em termos de **sentenças** que mudam o estado do programa
     3. Define sequências de comandos para o computador executar
     4. Semelhante a uma linguagem oral imperativa:
        + Chefe: - Some dois números!
        + Chefe: - Exiba o resultado!
        + Chefe: - Volte ao seu trabalho anterior!
        + Chefe: - etc.
     5. Relembrando: Estado de um programa é definido pelas suas estruturas de dados e variáveis
  2. [procedural](https://en.wikipedia.org/wiki/Procedural_programming) – existência de rotinas, sub-rotinas, ou funções para agrumapento de código.
  3. [declarativa](https://en.wikipedia.org/wiki/Declarative_programming) - não especifica a ordem em que as operações são executadas, mas a existência e condições para execução
  4. [não-estruturada](https://en.wikipedia.org/wiki/Non-structured_programming) - sequências de comandos ordenados
     1. Constituída por uma sequência de comandos ordenados;
     2. Comandos podem ser identificados por rótulos (labels) ou por número de linhas, permitindo saltos;
     3. Primeiras linguagens de programação;
     4. Desvantagens:
        + Código difícil de ler;
        + Não recomendada para grandes projetos;
        + Apenas tipos de dados básicos;
     5. Vantagens:
        + Em determinadas situações pode ser necessária para elevado desempenho;
        + Mais próximas às linguagens da máquina;
     6. **Exemplo: BASIC (1964)**
  5. [e](https://en.wikipedia.org/wiki/Object-oriented_programming)[struturada](https://en.wikipedia.org/wiki/Structured_programming) - diferentes estruturas para controle do fluxo do programa
     1. Criada para padronizar e melhorar a clareza, qualidade e tempo de desenvolvimento de programas de computadores;
     2. [por blocos](https://en.wikipedia.org/wiki/Block_(programming)) - definições de escopo de código por delimitadores
     3. [orientada a objeto](https://en.wikipedia.org/wiki/Object-oriented_programming)  - agrupa código junto com os dados que o código modifica