Paradigma de Programação

Programação Procedural ou Top Down:
- Baseia-se na ideia de que o programa é uma sequência de instruções ou procedimentos que manipulam dados.
- Enfatiza funções ou procedimentos que operam sobre dados.
- Os dados e as funções são separados; funções agem sobre os dados, mas não há uma conexão direta entre eles.

Programação Orientada a Objetos (POO)
- Organiza o software em "objetos" que combinam dados e comportamentos.
- Baseado na modelagem de problemas no mundo real, agrupando dados e comportamentos que atuam sobre esses dados em uma única unidade.

Conceitos em POO

Classe
  - É uma estrutura que define um tipo de objeto.
  - Serve como um molde para criar objetos, especificando quais dados e comportamentos eles terão.
  - Exemplo: Uma classe `Carro` pode definir atributos como `cor` e `modelo`, e métodos como `acelerar()` e `frear()`.

Objeto
  - É uma instância de uma classe.
  - Representa uma entidade concreta que possui estado (dados) e comportamento (métodos).
- Exemplo: Um objeto `meuCarro` pode ser uma instância da classe `Carro`, com atributos específicos como `cor = "vermelho"` e `modelo = "Fusca"`.

Encapsulamento
  - Refere-se ao conceito de esconder os detalhes internos de uma classe e expor apenas o que é necessário para o uso dos objetos.
  - Protege o estado interno do objeto e permite acesso controlado através de métodos públicos.
  - Exemplo: Os atributos `cor` e `modelo` da classe `Carro` podem ser privados, acessados e modificados apenas por métodos públicos da classe, como `setCor()` e `getModelo()`.

Abstração
  - Permite representar conceitos complexos por meio de classes e objetos, ocultando detalhes desnecessários.
  - Facilita o trabalho com ideias de alto nível sem se preocupar com a implementação detalhada.
  - Exemplo: A classe `Carro` abstrai a complexidade de como um carro é construído, permitindo que se trabalhe com o conceito de um carro sem precisar saber os detalhes da engenharia.

Herança
  - Permite que uma nova classe herde atributos e métodos de uma classe existente.
  - Facilita a reutilização de código e a criação de uma hierarquia de classes.
- Exemplo: Uma classe `CarroEsportivo` pode herdar de `Carro`, adquirindo todos os atributos e métodos de `Carro` e podendo adicionar novos atributos ou métodos específicos.

Polimorfismo

- Permite que diferentes classes implementem métodos com o mesmo nome de maneiras diferentes.
- Facilita a substituição e extensão de comportamentos sem alterar o código que usa esses métodos.
- Exemplo: Um método `desenhar()` pode ser definido em uma classe `Forma` e ser implementado de maneira diferente em classes derivadas como `Círculo` e `Quadrado`.

Como a POO Ajuda na Construção de Melhores Soluções e Software

Modularidade e reutilização

- A POO promove a criação de componentes modulares (classes) que podem ser reutilizados em diferentes partes do sistema ou em diferentes projetos.

Escalabilidade

- Permite a construção de sistemas mais complexos e escaláveis, com uma estrutura clara e organizada.

Desenvolvimento colaborativo

- Facilita o trabalho em equipe, onde diferentes desenvolvedores podem trabalhar em diferentes classes e objetos sem causar conflitos.