SISTEMAS DIGITAIS PARA MECATRÔNICA

Aluna: Geovanna Lissa da Silva Prado

Número de matrícula: 12211EMT022

Uberlândia

Março de 2025

1. Paradigma de Programação:

- Programação Procedural ou Top Down:
- 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;

2. 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;

3. 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 `Moto` pode definir atributos como `cor` e `modelo`, e métodos como `acelerar` e `frear`.

4. Objeto:

- É uma instância de uma classe;
- Representa uma entidade concreta que possui estado (dados) e comportamento (métodos);

Exemplo: Um objeto `minhaMoto` pode ser uma instância da classe 'Moto', com atributos específicos como `cor = "vermelho"` e `modelo = "Biz"`.

5. 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 `Moto` podem ser privados, acessados e modificados apenas por métodos públicos da classe, como `setCor` e `getModelo`.

6. 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 `Moto` abstrai a complexidade de como uma moto é construída, permitindo que se trabalhe com o conceito de um carro sem precisar saber os detalhes da engenharia.

7. 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 `MotoEsportiva` pode herdar de `Moto`, adquirindo todos os atributos e métodos da `Moto` e podendo adicionar novos atributos ou métodos específicos.

8. 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`.

9. 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.

10. Escalabilidade:

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

11. Desenvolvimento colaborativo:

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