D S T Q Q S S

1 / 1 / 22

POO

A programação orientada a objetos surgiu com o intuito de aproximar o manuseio das estruturas de um programa ao manuseio das coisas do mundo real. Esse novo paradigma se baseia em dois conceitos chave, classes e objetos. Todos os outros conceitos são construídos em cima desses dois.

Classe: é um conjunto de características e comportamentos que definem o conjunto de objetos pertencentes a essa classe.

Objeto: é uma instância de uma classe, então se tivermos uma classe carro, com seus atributos (motor, cambio, portas) e métodos (acelerar, frear, desligar), o objeto Astra é uma instancia da classe carro, assim como o Lancer ou demais carros, o que muda é o atr

Abstração, encapsulamento, herança e polimorfismo.

1° Encapsulamento: Ainda no exemplo do carro, sabemos que existem atributos que por sua vez podem ser definidos como publicos e privados, sendo os métodos e atributos públicos, aqueles que podem ser vistos e alterados independente do escopo da aplicação, como a cor de um carro, modelo de rodas etc, e os métodos e atributos privados, que não podem ser modificados fora do escopo da classe, pois pode causar mal func. no obj

* Os getters e setters entram aqui.

1 / 1 / 22

D S T Q Q S

2° Herança: Por herança, podemos fazer uma analogia: Existem as classes lancer e ive, sendo a classe lancer, a classe mãe de ive, onde ive vai herdar as características de lancer, porém com algumas alterações em seus atributos.

Quando dizemos que uma classe A é um tipo de classe B, dizemos que a classe A herda as características da classe B e que a classe B é a mãe da classe A, estabelecendo então uma relação de herança entre elas.

3° Polimorfismo: Ainda no exemplo dos carros, podemos dizer o seguinte, o astra, toca musica via pendrive, enquanto o lancer, toca música via Bluetooth.

Dizemos que o método tocarMusica é uma forma de polimorfismo, pois dois objetos, de classes diferentes, têm um mesmo método que é implementado de formas diferentes, ou seja, um método possui varias formas, varias implementações diferentes em classes diferentes, mas que possuem o mesmo efeito, daí o nome (polimorfismo)

Design Patterns: Alguns problemas aparecem com tanta frequência em POO que suas soluções se tornaram padrões de design de sistemas e modelagem de código voltado a objeto, a fim de resolve-los. Esses padrões de projeto nada mais são que formas padronizadas de resolver problemas comuns em linguagens OO

* kiss = Keep it simple, stupid
* dry = dont repeat yourself.

D S T Q Q S S

4/1/22

Interfaces: Muitos dos métodos dos carros são comuns em vários automóveis. Tanto um carro quanto uma motocicleta são classes cujos objetos podem acelerar, parar, acender o farol etc, pois são ações comuns a automóveis. Podemos dizer então que ambas as classes carro e moto são automóveis.

Quando duas (ou mais) classes possuem comportamentos comuns que podem ser separados em uma outra classe, dizemos que a "classe comum" é a interface, que pode ser "herdada" pelas outras classes, entre aspas, porque uma interface não é exatamente uma classe, mas sim um conj. de métodos que todas as classes que herdarem dela devem possuir (implementar).

Portanto, uma interface não é "herdada" por uma classe, e sim implementada.

Arquiteturas de software

MVC → Model, View, Controller

• Model Camada que possui a lógica da aplicação, responsável pelas regras de negócios, persistência com o BD e as classes de entidades. O model recebe requisições vindas do controller e gera respostas a partir dessas requisições.

Sempre que pensar na model, pense que ela terá conhecimento apenas dos dados armazenados no sistema e na lógica sobre eles, por se tratar do núcleo da aplicação. É na model também que o crud deve ser realizado.