**Orientação à Objetos**

**Tipos Abstratos de Dados (TAD)**

TAD são formas de representar novos tipos de dados através da Programação Orientada a Objetos. Esses tipos abstratos de dados (TAD) podem servir de estruturas de dados. São também formados como um conjunto de dados por um ou mais tipos de dados já existentes (strings, inteiros, booleanos). Desta forma é possível criar algo mais complexo. Cada tipo de dado inserido é chamado de atributo. Podem também serem formados por funções que são chamadas de métodos.

Através dos TADs podemos mapear de forma computacional as entidades (reais ou abstratas) que fazem parte de um problema que se deseja resolver. Desta forma conseguimos ter uma visão mais concreta do problema, onde ao ser utilizado o mesmo vocabulário do processo real existem assim no código entidades compatíveis com a realidade do problema.

Por exemplo necessitamos criar um código que armazena contatos onde serão guardados nome, telefone e e-mail. Desta forma criamos uma entidade chamada “Agenda” (TAD) que irá armazenar uma lista com outras entidades chamadas “Contato” (TAD). Dentro de Agenda teremos métodos que permitam adicionar, remover, buscar Contatos, entre outros. Desta forma centralizamos todas as informações de Contatos em uma única entidade, impedindo assim que as informações se misturem ou se percam. Na entidade “Agenda” centralizamos então todas as operações que o sistema pode fazer. É mais fácil falar de uma entidade chamada “Agenda” que possua uma lista de “Contatos” do que falar de toda essa informação espalhada por vários vetores diferentes.

A principal função da Programação Orientada a Objetos é exatamente isso, torar o código o mais próximo da vida real possível, estruturando processos com os mesmos nomes utilizados na vida real e dividindo a responsabilidade entre diferentes entidades para não centralizar todo o código em um único programa enorme que faz tudo.

**Criando Objetos**

Normalmente é necessário criar uma classe antes de criar objetos. Porém por Javascript ser dinâmica, suportar POO e por não ser o paradigma principal da linguagem é possível a criação de objeto sem criar classes. Isso facilita na criação de objetos únicos que são usados apenas em lugares determinados no código, não os reaproveitando em outras partes do sistema. Exemplos de criação de objetos sem criação de classes:

**const** pessoa1 = { nome : "Carlos", idade : 20 };

**const** quadrado = {

    base : 10,

    altura : 10,

    calcularArea : **function**() **=>** { return this.base \* this.altura; }

};

Para se referir a um atributo ou método dentro do objeto é necessário a utilização do this para que ser entendido que é algo que pertence ao objeto e não uma variável declarada fora dele. Pode-se criar objetos de dentro de outro objeto também.

Para que o objeto seja reutilizável, ou seja, capaz de ser utilizado em outros pontos do código, se faz a necessário a utilização de classes.

Exemplos de atributos (propriedades) : carro -> cor, marca, modelo, ano de fabricação, etc.

Exemplos de métodos (coisas que o objeto faz) : carro -> ligar, desligar, ligar rádio, correr, parar, etc.

**Bibliografia**