**1. DEFINIÇÃO DO PARADIGMA DE ORIENTAÇÃO A OBJETOS**

“Programação orientada a objetos (POO) é um paradigma que viabiliza o desenvolvimento de aplicações fazendo uso do modelo orientado a objetos. Entende-se como modelo orientado a objetos o conjunto de "coisas" que fazem parte do contexto atual em estudo, e normalmente os objetos encontrados na análise se agrupam em classes. Estas classes, em conjunto com as associações entre elas, definem a estrutura do sistema em estudo. Uma das principais características da POO é a capacidade de reutilização, ou seja, de otimização da produtividade, aumento de qualidade, diminuição de tempo e custos de manutenção. Todas essas características dependem da maneira como o software é desenvolvido. Muitos dos programadores profissionais reutilizam códigos, porém o perfeito reúso consiste no uso completo de um código gerado para algum sistema, sem a necessidade de qualquer outra adaptação. Outro fator considerado como vantagem é a manutenibilidade, ou seja, a facilidade na manutenção dos projetos. Este fator depende de como o sistema foi estruturado e das técnicas de programação que foram usadas ao longo do desenvolvimento”. (ARAUJO, 2017).

“O Paradigma Orientado a Objeto (POO) tem como principal característica uma melhor e maior expressividade das necessidades do nosso dia a dia, ele possibilita criar unidades de código mais próximas da forma como pensamos e agimos, facilitando o processo de transformação das necessidades diárias para uma linguagem orientada a objetos. É válido ressaltar que em alguns casos, somente utilizar uma linguagem orientada a objetos não garante que se esteja programando efetivamente orientado a objetos. Isso pode ocorrer devido a "vícios estruturados" dos novos programadores desse paradigma, ou mesmo de programadores experientes, mas que no final terminaram cometendo o mesmo erro: programar estruturado em uma linguagem OO.

Além do que já foi citado, há ainda uma outra característica, talvez filosófica, sobre esse paradigma de programação. Ele defende que é possível representar todo e qualquer processo do mundo real a partir da utilização de apenas três estruturas básicas:

\*Sequência - Os passos devem ser executados um após o outro, linearmente. Ou seja, o programa seria uma sequência finita dos passos. Em uma unidade de código, todos os passos devem ser feitos para se programar o algoritmo desejado.

\*Decisão - Uma determinada sequência de código pode ou não ser executado. Para isto, um teste lógico deve ser realizado para determinar ou não sua execução. A partir disto, verifica-se que duas estruturas de decisão (também conhecida como seleção) podem ser usadas: a if-else e a switch.

\*Iteração - É a execução repetitiva de um segmento (parte) do programa. A partir da execução de um teste lógico, a repetição é realizada um número finito de vezes. Estruturas de repetição conhecidas são: for, foreach, while, do-while, repeat-until, entre outras (dependendo da linguagem de programação).” (CARVALHO, 2016).

**2. COMPARAÇÕES COM O PARADIGMA ESTRUTURADO**

| **Programação orientada a objetos** | **Programação estruturada** |
| --- | --- |
| Métodos | Procedimentos e funções |
| Instâncias de variáveis | Variáveis |
| Mensagens | Chamadas a procedimentos e funções |
| Classes | Tipos de dados definidos pelo usuário |
| Herança | Não disponível |
| Polimorfismo | Não disponível |

“Na Programação estruturada observamos algumas vantagens como um controle mais eficaz quanto ao fluxo de execução do programa e a facilidade de compreender o código quando o mesmo é analisado. Como principal desvantagem temos a facilidade em desenvolver um código confuso caso o desenvolvedor faça o tratamento dos dados juntamente com o restante da execução do programa, além do que o reuso de código se torna um pouco complicado devido a não definição da tarefa. Já na programação orientada a objetos temos como vantagens a reutilização de código e a melhor organização do código do programa. Porém, alguns detalhes tornam essa programação um pouco prejudicada quando comparada a programação estruturada como, por exemplo, o desempenho do código e a confusão na hora de aplicar os conceitos de orientação a objetos.

Portanto, não podemos assumir que um tipo de programação é mais vantajoso que o outro, pois em ambos existem características bem peculiares nas suas definições, deixando cada uma mais viável de ser implementada, dependendo do negócio escolhido, para facilitar o bom entendimento e a manutenibilidade futura do código feito.” (DEVMEDIA).

**3. DEFINIÇÃO DE:**

**3.1 Classe**

“Classe é um conjunto ou uma categoria de objetos que tem propriedades e métodos. Na realidade, é a implementação de características específicas desse conjunto de objetos e dos comportamentos que realiza algo específico já programado.” (CARVALHO, 2016).

“Classe é uma estrutura que abstrai um conjunto de objetos com características similares. Uma classe define o comportamento de seus objetos através de métodos e estados possíveis destes objetos através de atributos. Em outros termos, uma classe descreve os serviços providos por seus objetos e quais informações eles podem armazenar. Logo na primeira linha, aparece o seguinte texto: "abstrai um conjunto de objetos com características similares". Ou seja, o objetivo de uma classe é definir, servir de base, para que futuramente será o objeto. É através dela que criamos o "molde" aos quais os objetos deverão seguir. Este "molde" definirá quais informações serão trabalhadas e como elas serão manipuladas”.

**3.2 Atributo**

“Propriedades ou atributos consistem de características atribuídas aos objetos. Em uma janela, por exemplo, tem-se material, cor e tipo de vidro, dentre outros. Outro exemplo é um livro, que possui título, autor, editora e ISBN. Já uma expressão matemática normalmente é composta por dois operandos e um operador”."Atributo' é o elemento de uma classe responsável por definir sua estrutura de dados. O conjunto destes será responsável por representar suas características e fará parte dos objetos criados a partir da classe. Assim como nas classes, os atributos podem ser representados a partir de substantivos. Além destes, podemos também usar adjetivos. Pensar em ambos pode facilitar o processo de identificação dos atributos”.(CARVALHO, 2016).

**3.3 Método**

“Referem-se às ações (operações) aplicadas por um objeto ou suas reações. Na janela, por exemplo, o abrir e fechar são vistos como comportamentos, que podem disparar eventos que poderíamos descrever com "ao abrir ou fechar a janela". Em um carro, dar partida, acelerar, frear ou trocar de marcha são métodos de ação. Os comportamentos e eventos mapeados para uma linguagem são conhecidos como implementação dos métodos”.

“Método é uma porção de código (sub-rotina) que é disponibilizada pela classe. Esta é executada quando é feita uma requisição a ela. Um método serve para identificar quais serviços, ações, que a classe oferece. Eles são responsáveis por definir e realizar um determinado comportamento.” (CARVALHO, 2016).

**3.4 Objeto**

“Objeto é qualquer estrutura modular que faz parte de algo. Em nossa vida, em nosso cotidiano, lidamos constantemente com objetos, alguns físicos (como uma janela) e outros conceituais (como uma expressão matemática). Uma janela, por exemplo, é um objeto de uma casa, de um carro ou de um software com interface gráfica para o usuário. Pense também em um livro como parte de uma biblioteca. Outro exemplo pode ser uma equação matemática composta por várias expressões. Cada objeto possui propriedades, comportamentos e métodos que identificam e o diferenciam dentre outros objetos semelhantes.” (CARVALHO, 2016).

“No paradigma orientado a objetos, temos o objeto como elemento principal. Aliás, a programação orientada a objetos surge da necessidade de termos uma linguagem de programação cujo código fosse escrito o mais próximo possível da maneira como os seres humanos interagem. Os objetos representam qualquer "coisa" do mundo real: um carro, uma cadeira, um animal e até mesmo uma pessoa.” (OBERLEITNER, MASIERO, 2021)

**4. O QUE SIGNIFICA INSTÂNCIA DE UM OBJETO**

“A partir de uma classe, podemos criar tantos objetos quanto desejarmos. Por exemplo, podemos instanciar o objeto B1 (que seria o seu boletim) e B2 (que seria o boletim do seu amigo de sala). Cada um seria um boletim diferente, com comportamento e estados próprios.’ (OBERLEITNER, MASIERO, 2021)

“Para identificar e nomear os objetos, devemos seguir o mesmo principio das classes: pensar em substantivos. Isso ocorre porque um objeto é criado a partir de uma classe. Por fim, o código a seguir ilustra como criar um a partir da classe Personagem.

Personagem personagem = new Personagem ();

Esse código usa o construtor que foi definido na classe Personagem para criar (instanciar) um objeto a partir dela. Com o auxílio do operador New, um objeto do tipo Personagem é instanciado e armazenado na variável personagem.” (CARVALHO, 2016).

**5. MODIFICADORES DE ACESSO**

**5.1 public**

“De acordo com Horstimann (2019), o modificador de acesso public permite o acesso de qualquer objeto ao código interno da classe. É a forma menos segura de acesso.” (OBERLEITNER, MASIERO, 2021).

“Membros declarados como public poderão ser acessados livremente a partir da própria classe em que foram declarados, a partir de classes descendentes e a partir do programa que faz uso dessa classe (manipulando o objeto em si). Na UML, simbolizamos com um caractere "+" na frente da propriedade.” (Dall'Oglio, 2018).

**5.2 private**

“O modificador de acesso private permite o acesso somente dentro da classe em que o elemento foi criado.” (OBERLEITNER, MASIERO, 2021).

“Membros declarados como private somente podem ser acessados dentro da própria classe em que foram declarados. Não poderão ser acessados a partir de classes descendentes nem a partir do programa que faz uso dessa classe (manipulando o objeto). Na UML, simbolizamos com um caractere "-" na frente da propriedade.” (Dall'Oglio, 2018).

**5.3 protected**

“Membros declarados como protected somente podem ser acessados dentro da própria classe em que foram declarados e a partir de classes descendentes, mas não poderão ser acessados a partir do programa que faz uso dessa classe (manipulando o objeto em si). Na UML, simbolizamos com um caractere "#" na frente da propriedade.” (Dall'Oglio, 2018).

**Referências:**

CARVALHO, Thiago. **Orientação a Objetos: Aprenda seus conceitos e suas aplicabilidades de forma efetiva.** Editora Casa do Código: São Paulo, 2016.

ARAUJO, Everton Coimbra. **Orientação a Objetos em C#: Conceitos e implementações em .NET**. Editora Casa do Código: São Paulo, 2017.

Programação Orientada a Objetos e Programação Estruturada. <**https://www.devmedia.com.br/programacao-orientada-a-objetos-e-programacao-estruturada/32813>**

OBERLEITNER, Allen; MASIERO, Andrey. **Programação orientada a objetos Allen Obertleiner.** Editora Senac: São Paulo, 2021.

Dall'Oglio, Pablo. **PHP Programando com Orientação a Objetos**. Novatec, 4ª edição, 2018: São Paulo.