**Programação Orientada à Objetos – (POO)**

A Programação Orientada à Objetos é atualmente um dos paradigmas mais utilizados por empresas e desenvolvedores no mundo. Esse modelo utilizado para o desenvolvimento de software revolucionou e facilitou a vida dos programadores em 101%.

A orientação à objetos é um paradigma que possui 4 pilares fundamentais, sendo eles:

* **Abstração**
* **Encapsulamento**
* **Herança**
* **Polimorfismo**

**Abstração**

A **abstração** na orientação à objetos é onde tudo começa, é o ponto chave para que a abstração do mundo real seja transformado em um sistema. No caso da orientação à objeto, as abstrações são adotadas no formato de “classes”.

Uma **classe** é uma abstração de um objeto do mundo real para o software, uma classe possuí atributos e métodos. O seguinte exemplo abaixo irá mostrar como funciona o processo de abstração:

Vamos considerar o seguinte problema:

1. Preciso criar uma classe que represente a abstração de um funcionário de uma empresa.
2. Preciso definir quais são os atributos e métodos dessa classe que representará o meu funcionário.

Um **funcionário** possui as seguintes características, que na orientação à objetos conhecemos como ‘atributos’: nome, idade, profissão, rg, cpf e entre outros.

Um **funcionário** pode executar as seguintes funções: marcar ponto, trabalhar, descansar e etc.

Essa é abstração de um **funcionário** passada para a orientação à objetos.

**Encapsulamento**

O encapsulamento é uma das principais técnicas que define a programação orientada a objetos. Se trata de um dos elementos que adicionam segurança à aplicação em uma programação orientada a objetos pelo fato de esconder as propriedades, criando uma espécie de caixa preta.

A maior parte das linguagens orientadas a objetos implementam o encapsulamento baseado em propriedades privadas, ligadas a métodos especiais chamados getters e setters, que irão pegar e alterar os valores de uma propriedade, respectivamente. Essa atitude evita o acesso direto a propriedade do objeto, adicionando uma outra camada de segurança à aplicação.

Para fazermos um paralelo com o que vemos no mundo real, temos o encapsulamento em outros elementos. Por exemplo, quando clicamos no botão ligar da televisão, não sabemos o que está acontecendo internamente. Podemos então dizer que os métodos que ligam a televisão estão encapsulados.

**Exercícios**

1. Crie uma classe com o nome de “Funcionário” e implemente os atributos e métodos citados no diagrama acima.

2 – Após a criação da classe, no programa principal faça a instanciação de objetos da classe “Funcionário” e altere os seus atributos e faça a chamada dos seus métodos.

**Herança**

O reuso de código é uma das grandes vantagens da programação orientada a objetos. Muito disso se dá por uma questão que é conhecida como herança. Essa característica otimiza a produção da aplicação em tempo e linhas de código.

Para entendermos essa característica, vamos imaginar uma família: a criança, por exemplo, está herdando características de seus pais. Os pais, por sua vez, herdam algo dos avós, o que faz com que a criança também o faça, e assim sucessivamente. O objeto abaixo na hierarquia irá herdar características de todos os objetos acima dele, seus “ancestrais”. A herança a partir das características do objeto mais acima é considerada herança direta, enquanto as demais são consideradas heranças indiretas. Por exemplo, na família, a criança herda diretamente do pai e indiretamente do avô e do bisavô.

A imagem acima mostra um ciclo de **herança**, vamos considerar que avô, pai e neto são classes, e que todas elas possuem métodos e atributos. Analisando a sequência do diagrama da direita para a esquerda, podemos analisar que o neto herda atributos e métodos tanto de seu pai e seu avô, e seu pai herda de seu avô.

**Exercícios**

1. Crie uma classe abstrata com o nome de “Eletrônico”, uma classe com o nome de “Rádio”, uma classe com o nome de “Televisão” e implemente os atributos e métodos base na classe “Eletrônico”. Após, herde os mesmos nas classes “Rádio” e “Televisão” e crie atributos e métodos específicos para cada uma destes. Finalizada a construção dessas classes, no programa principal, instancie objetos dessas classes e faça a alteração dos atributos e a chamada dos métodos.

**Polimorfismo**

Outro ponto essencial na programação orientada a objetos é o chamado polimorfismo. Na natureza, vemos animais que são capazes de alterar sua forma conforme a necessidade, e é dessa ideia que vem o polimorfismo na orientação a objetos. Como sabemos, os objetos filhos herdam as características e ações de seus “ancestrais”. Entretanto, em alguns casos, é necessário que as ações para um mesmo método seja diferente. Em outras palavras, o polimorfismo consiste na alteração do funcionamento interno de um método herdado de um objeto pai.

**Exemplo:**

Suponha que temos uma classe chamada “Aparelho Eletrônico”, nós sabemos existem vários tipos de aparelhos eletrônicos, certo? Um aparelho eletrônico pode ser um rádio, televisão, celular, câmera fotográfica e entre outros.

Agora, voltando um pouco atrás onde falamos que uma classe possui métodos e atributos, vamos considerar que a classe “Aparelho Eletrônico” seja a nossa classe genérica, ou seja: um aparelho eletrônico pode assumir diversas formas, das quais foram mencionadas anteriormente.

Agora, vamos definir que a nossa classe “Aparelho Eletrônico” possui um método chamado “**ligar**”. Como foi analisado antes, um aparelho eletrônico pode assumir diversas formas diferentes, e todos eles necessitam ser **ligados** para funcionar, certo? Mas nem todos eles são ligados da mesma forma. Exemplo: a forma de ligar um rádio pode ser a de ligar uma câmera fotográfica e entre outros, mas todos eles executam o método ligar.

Graças à esse tipo de abordagem, podemos reutilizar o método **ligar** da nossa classe genérica “Aparelho Eletrônico” em classes que assumem a forma de um aparelho eletrônico.

**Exercícios**

1. Na classe “Eletrônico” implementada anteriormente, crie um método abstrato chamado “EmitirSom” e o sobrescreva nas classes “Rádio” e “Televisão”.