# Orientação a Objetos

## Material de Apoio

# Introdução

Orientação a objetos nos permite de forma mais clara e objetiva a representação de um objeto do mundo real em um programa digital, fornecendo comportamentos e atributos semelhantes aos verdadeiros *“a orientação a objetos traz um enfoque diferente da programação estruturada, no sentido de adotar formas mais próximas do mecanismo humano para gerenciar a complexidade de um sistema” (Rocha Douglas, 2009).* A orientação a objetos traz em seu paradigma, o conceito de reutilização de seu código a fim de prover menos redundância de código proporcionando menos manutenção e correções mais assertivas.

Para entender como programar orientado a objetos necessitamos entender o paradigma da orientação a objetos criado na década de 70, diferente de outras abordagens de desenvolvimento como a programação Estruturada ou Funcional, Simplificando o código e representando o ambiente físico no virtual.

# Paradigmas de Programação

## Programação Estruturada

Este paradigma tem como base o conceito de que todo programa pode ser reduzido em até três estruturas: sequência, decisão e iteração.

- Sequência: Instruções executadas uma por vez em que aparecem no programa
- Decisão: Instruções condicionais que introduzem lógica ao programa.
- Iteração: Instruções executadas várias vezes, conhecidas como loopings.

### Exemplo de Linguagens

- C;
- Basic;
- Pascal;

### Vantagens desse modelo

Os problemas podem ser representados em uma única estrutura podendo ser quebrado em sub algoritmos, podendo ser invocados na estrutura.

### Desvantagens desse modelo

Os dados neste modelo são separados das funções, mudanças nessas estruturas acarretam em manutenção por todo o código para adequação da alteração.

## Programação Orientada a Objetos

Este paradigma é baseado na decomposição e interação dos elementos de um ambiente real, em unidades de software denominados objetos, Cada objeto é criado de uma classe composta por atributos, propriedades e métodos.

### Exemplo de Linguagens

- C++
- C#
- Java
- Object Pascal
- Python

### Vantagens desse modelo

A principal vantagem deste paradigma é que alterações nestas pequenas unidades de código, não devem influenciar na solução inteira podendo ser reaproveitadas em outros pontos da solução.

### Desvantagens desse modelo

Por se tratar de uma forma mais complexa de visualizar o software por muitas vezes esta abordagem não é adotada, ou adotada de maneira errônea podendo ao invés de trazer vantagens ao software podendo ser um ponto de quebra no sistema.

# Objeto e Classe

Em orientação a objetos tudo é considerado um objeto, para que seja possível a interação entre as unidades, um objeto sempre será criado a partir de uma classe, a classe contém todas as definições para que o objeto possa existir e interagir com o software.

## Classe

Para encontrarmos as classes de um ambiente em estudo para desenvolvimento de um software, analisamos as entidades que interagem neste ambiente, existem várias técnicas para análise de um ambiente em orientação a objetos mas o mais comum é a metodologia de Use Cases(Casos de Uso), esta metodologia separa o ambiente em dois elementos sendo eles Atores e Casos de Uso, Os atores representam entidades do ambiente que trocam informações entre si, já os casos de uso representam as ações desta entidade.

A Partir da abordagem acima podemos então obter as entidades que irão compor este ambientes, chamadas de classes estas entidades classificam um grupo que

possua os mesmos comportamentos e propriedades, a fim de tratar de forma comum a interação destes elementos com o sistema.

Imagine que para construir um carro precisamos de um projeto deste carro, neste projeto está contido todos os atributos que o carro deve possuir (Tipo do Motor, Tipo dos Freios, Carroceria, etc..), e seus comportamentos (Acelerar, Frear, Acionar Palhetas, etc), com estes dados podemos criar um carro, a classe é o projeto do objeto com base na classe teremos a estrutura do objeto após criado.

### Objeto

Podemos dizer que o objeto é a materialização da classe a partir do momento em que criamos um objeto baseado em uma classe, este objeto está criado na memória do computador e podemos acessá-lo para atribuir valores a sua estrutura e utilizar seus comportamentos. Seguindo o exemplo anterior de um projeto podemos criar N carros, porém eles serão diferentes em alguns aspectos (Cor, Opcionais, etc), e mesmo assim ainda derivam da mesma classe.

## Pilares da Orientação a Objetos

Os pilares da orientação a objetos são quatro definidos como Abstração, Encapsulamento, Herança e Polimorfismo, não é possível criar um ambiente orientado a objetos sem adotar estas quatro premissas pois o poder da orientação a objetos deriva destes quatro conceitos.

| Orientação a Objetos | | | |
|---|---|---|---|
| Abstração | Encapsulamento | Herança | Polimorfismo |

### Abstração

A abstração é a primeira e mais importante fase de implementação de um software orientados a objetos, nesta fase o analista concentra-se em abstrair somente os conteúdos relevantes do ambiente real em estudo, a fim de trazer apenas dados relevantes para o sistema, “é a habilidade de concentrar-se nos aspectos essenciais de um contexto qualquer, ignorando características menos importantes ou acidentais. Em modelagem orientada a objetos, uma classe é uma abstração de entidades existentes no domínio do sistema de software.” (Centro Paula Souza).

Neste ponto definimos qual será a identidade a classe esta identidade deve ser única no sistema podendo ser identificado a classe por sua identidade, ao escolher a identidade da classe (Nome), devemos nomear com algo que realmente

represente a classe para que o código fique intuitivo e fácil de ler. nesta fase também definimos as propriedades e métodos da classe ainda de forma conceitual.

## Encapsulamento

Encapsulamento corresponde ao nível de segurança da classe para com o ambiente, por meio do encapsulamento expomos as demais classes que irão interagir com a entidade somente o que elas devem interagir sendo por regras de negócio do domínio ou comportamento da própria entidade, por meio de mecanismos da linguagem conseguimos expor de forma pública os métodos e propriedades que poderão ser acessados ou de forma privada sendo acessados somente pela classe.

Este paradigma é baseado na decomposição e interação dos elementos de um ambiente real, em unidades de software denominados objetos, Cada objeto é criado de uma classe composta por atributos, propriedades e métodos.

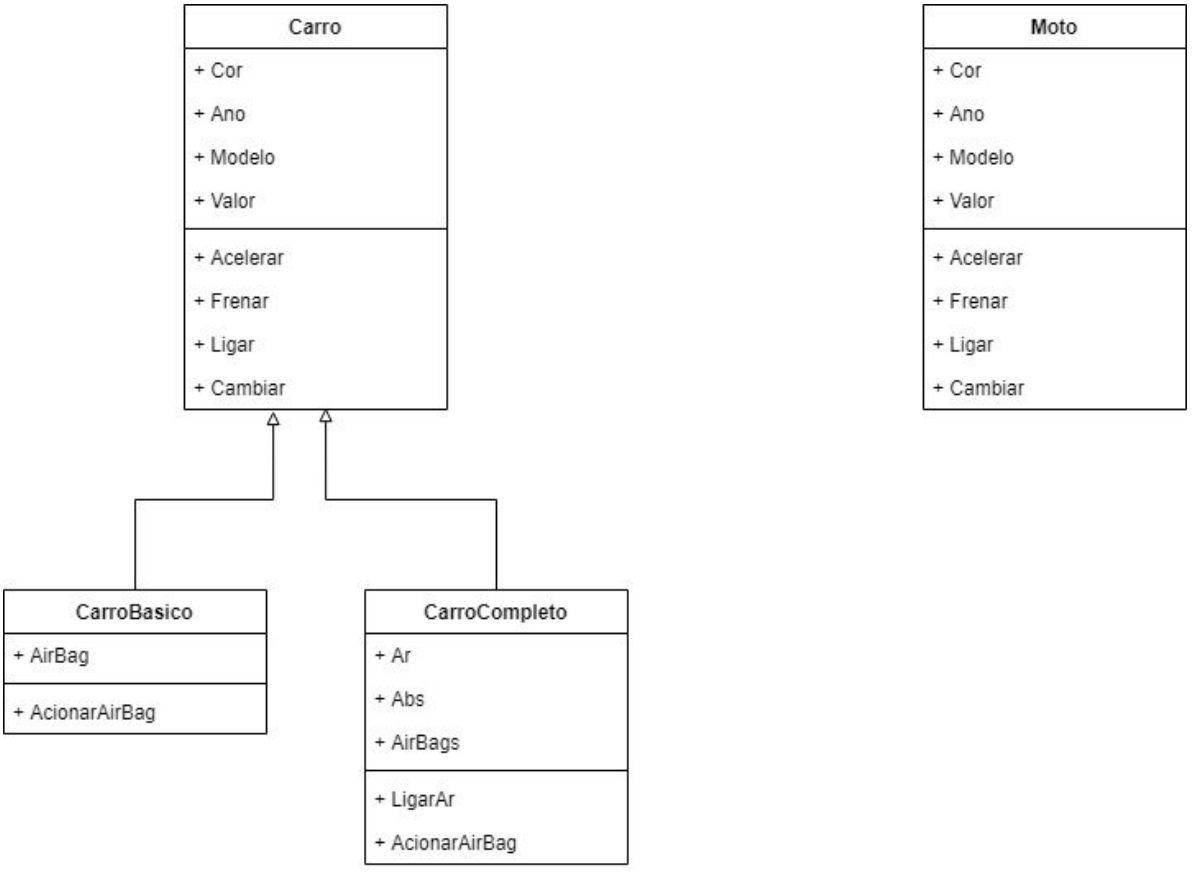
## Herança

Herança tem por sua finalidade fornecer um mecanismo em que seja possível reaproveitarmos o código anteriormente escrito em uma entidade SuperClasse (Pai) e recebermos seus comportamentos em suas SubClasses (Filho), desde que estas classes realmente estejam ligadas por uma relação hereditária. Como exemplo, quando temos um filho o mesmo pode herdar características dos pais e avós, porém não herdaria características de um outro parente por não possuir vínculo hereditário com o mesmo.

Em um exemplo mais aplicado a Orientação a objetos imagine um ambiente que controlamos o comportamento de automóveis, para este exemplo utilizaremos duas entidades base chamadas de Carros e Motos.

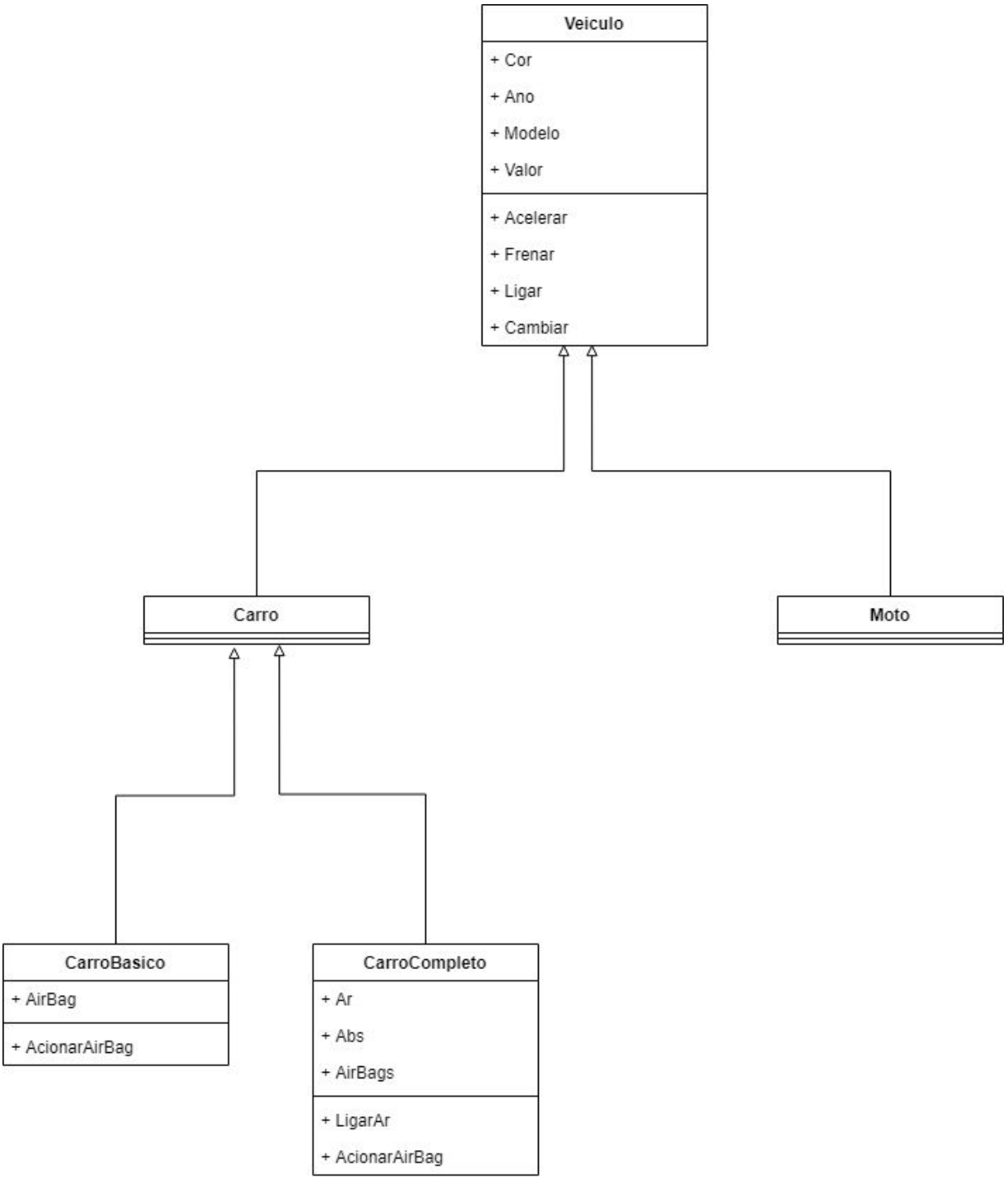

Estes objetos terão propriedades como Cor, Ano, Modelo, Valor e métodos representando ações dos objetos como Acelerar, Ligar, Frenar, Cambiar sendo representados da seguinte forma.


Estes objetos podem contar com sub objetos, para especializar ainda mais este ambiente se identificado na fase de abstração que o mesmo deva ser especializado até um nível específico, seguimos então especializando somente a entidade carro, e para esta entidade especializamos entre Carros Básicos e Carros Completos.




Note que ao especializar a classe temos novos atributos em cada classe, e também ações específicas mas ainda continuaremos tendo as ações base que recebemos por meio da herança. Como último passo deste ambiente notamos que a entidade carro e a entidade moto são exatamente iguais estruturalmente porém precisamos

desta divisão para o ambiente, então generalizamos também ambas estas entidades para um pai.

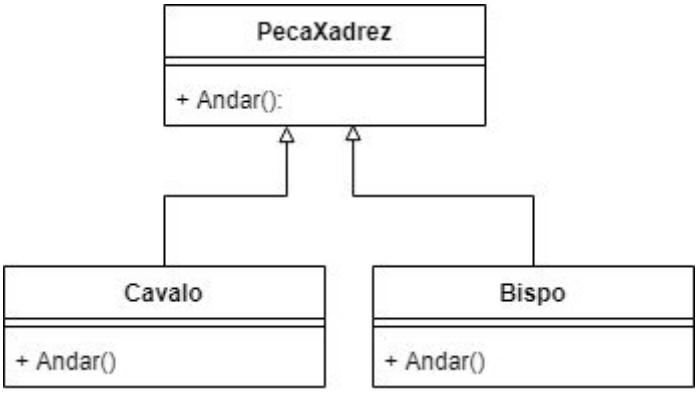



Nosso ambiente nesse momento permanece inalterado pois a única ação a ser feita foi a generalização onde , isolamos os comportamentos partilhados em uma superclasse para atender as necessidades do ambiente e prover reutilização do código já implementado.

## Polimorfismo

O polimorfismo é um pilar importante da orientação a objetos seu nome significa "várias formas”, este mecanismo atribui ao objeto a possibilidade de se comportar de várias formas pertencendo a uma mesma linha de herança, para podermos implementar o polimorfismo em OO contamos com um recurso chamado reescrita a reescrita consiste em alterar o comportamento do método por meio de reescrita literalmente do código herdado mantendo o comportamento base somado ao novo, ou simplesmente alterando todo o código e também contamos com um recurso denominado sobrecarga onde alteramos as características do método para utilização de mais uma possibilidade de método no código.

Como exemplo podemos pensar em uma entidade peça de Xadrez todas as peças possuem o método andar porém cada peça tem suas regras de deslocamento pelo tabuleiro podemos utilizar este recurso.




Neste exemplo apesar de visualmente não podemos ver a diferença os métodos andar contidos nas demais entidades serão reescritos, podendo ter seus comportamentos específicos, mas então onde está o polimorfismo está na capacidade de o desenvolvedor trabalhar com um único tipo de objeto declarado "Peça de Xadrez”, porém seu comportamento será alterado mediante a instância atribuída “Cavalo”, “Bispo”, “Torre” e etc..

# Referências

*Rocha Douglas, 2009 - Programação Java com Ênfase em orientação a objetos*

Centro Paula Souza - Codificando OOP com Visual Basic 2008 Express