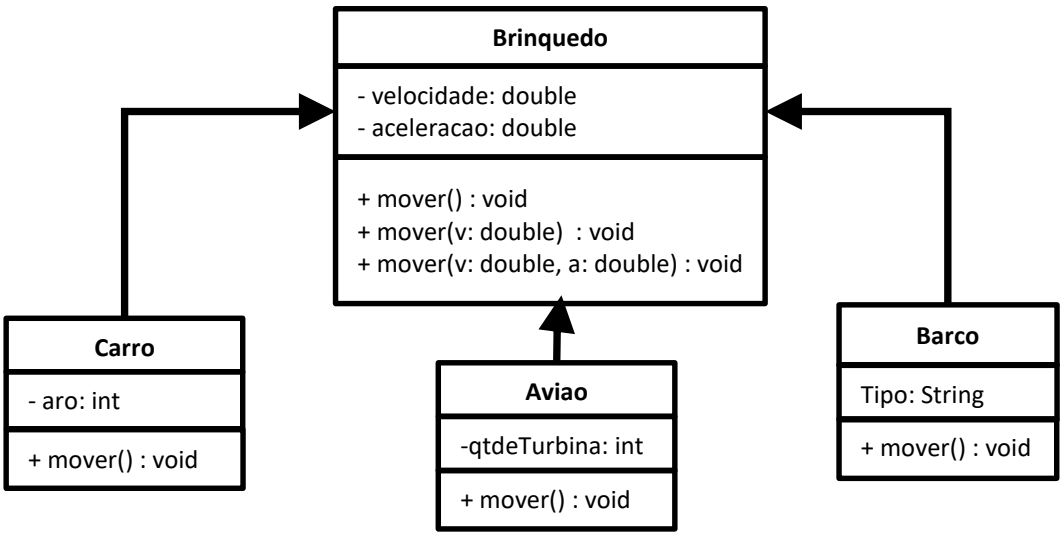
# Lista de Exercícios 04 – POO – Sobrecarga de Métodos, Sobrescrita de Métodos e Polimorfismo

**Instruções:**

- Enviar em um **ÚNICO** arquivo .ZIP ou .RAR contendo os projetos.
- Crie um ÚNICO projeto com o nome **Lista04** e para cada exercício prático (implementação) crie um novo pacote com o nome **ExercicioXX**.
- Só serão avaliados os exercícios enviados dentro do prazo.
- Esta Lista de Exercícios tem caráter **individual**.
- "*A resolução das Listas de Exercícios é a melhor forma de se preparar para a Prova! Os exercícios da Prova serão baseados nas Listas de Exercícios! Bom trabalho!!!*"

Implemente em C#, utilizando os conceitos de POO, os softwares descritos a seguir:

1. Escreva um programa em C# que implemente o Polimorfismo a partir do método mover(), presente na superclasse Brinquedo e nas subclasses Carro, Aviao e Barco, imprimindo na tela o nome da classe e o movimento. Ex.: "O avião voa." Crie pelo menos dois métodos construtores e encapsule os atributos. Por fim, crie uma classe principal, onde deverão ser criados os objetos das classes, além das entradas (*Console.ReadLine*) e saídas (*Console.WriteLine*) do programa.




2. Crie a classe Imovel, que possui um endereço e um preço. Crie os métodos de acesso e um método ImprimePreco(). Crie também uma classe Novo, que herda Imovel e possui um adicional no preço e crie os métodos de acesso e um método ImprimePreco(), acrescentando esse valor adicional. Crie uma classe Velho, que herda Imovel e possui um desconto no preço e crie os métodos de acesso e um método ImprimePreco(), concedendo esse desconto. Crie uma classe de Teste com o método *Main*. No método *man* crie um objeto da classe Imovel e peça para o usuário digitar

1 para novo e 2 para velho. Peça para o usuário digitar também o valor do imóvel e o valor adicional ou o valor do desconto, considerando o tipo de imóvel definido pelo usuário. Por fim, imprima o valor final do imóvel.

3. Crie uma classe chamada Ingresso que possui um valor em reais e um método imprimeValor(). Crie uma classe Normal, que herda Ingresso e possui um método que imprime o valor do ingresso e a mensagem "Ingresso Normal". Crie uma classe VIP, que herda Ingresso e possui um valor adicional. Crie um método que retorne o valor do ingresso, com o valor adicional incluído. Crie uma classe CamaroteInferior, que herda da classe VIP e possui a localização do ingresso e métodos para acessar e imprimir esta localização e para imprimir o valor do ingresso e a mensagem "Ingresso VIP - Camarote Inferior". Crie também uma classe CamaroteSuperior, que herda da classe VIP e é mais cara (possui mais um valor - adicionalSuperior). Esta última possui um método para retornar o valor do ingresso. Por fim, crie uma classe de Teste com o método *Main* e crie objeto da classe Ingresso. Peça para o usuário digitar 1 para normal e 2 para VIP. Conforme a escolha do usuário, diga se o ingresso é do tipo normal ou VIP. Se for VIP, peça para ele digitar 1 para camarote superior e 2 para camarote inferior. Conforme a escolha do usuário, diga se que o VIP é camarote superior ou inferior. Peça para o usuário digitar as informações necessárias para a instanciação do objeto ingresso, dependendo do tipo de ingresso (preço, adicional, adicionalSuperior). Por fim, imprima o valor do ingresso.

4. Implemente a classe Funcionario contendo os atributos nome, cpf e salario e os métodos de acesso e pelo menos dois métodos construtores. Implemente também uma classe Gerente, filha da classe Funcionario e que possui uma senha para identificação. Implemente os métodos de acesso e pelo menos dois métodos construtores. Todo fim de ano, os funcionários do nosso banco recebem uma bonificação. Os funcionários comuns recebem 10% do valor do salário e os gerentes, 15%. Crie um método chamado “GeraBonificação()” para gerar a bonificação para o Funcionário. Faça uma sobrescrita desse método na classe Gerente, gerando a bonificação correta para o mesmo.
   O cálculo da bonificação de um Gerente pode ser feito também a partir do acréscimo de um determinado valor em seu salário. Faça uma sobrecarga do método “GeraBonificação()” da classe Gerente, passando como parâmetro um determinado valor para a bonificação.
   Implemente também uma classe para controlar as bonificações, denominada “ControleDeBonificacoes”. Essa classe terá o atributo “totalDeBonificacoes” iniciando com o valor 0. Esse atributo armazenará o total de bonificações. Além disso, essa classe terá um método para registrar as bonificações (“RegistaBonifacoes”). Esse método receberá como parâmetro funcionário qualquer (independente do seu tipo) e

irá somar a bonificação desse funcionário ao total de bonificações. Deverá ser implementado um método para ver o total de bonificações.
Por fim, implemente uma classe para testar as classes implementadas anteriormente, contendo o método *Main*. Instancie os seguintes objetos e define os seguintes valores.

```
1. ControleDeBonificacoes controle = new
   ControleDeBonificacoes();
2. Funcionario funcionario1 = new Funcionario();
3. funcionario1.setSalario(5000.0);
4. controle. RegistaBonifacoes(funcionario1);
5. Gerente gerente1 = new Gerente();
6. gerente1.setSalario(1000.0);
7. controle.RegistaBonifacoes(gerente1);
8. System.out.println(controle.getTotalDeBonificacoes());
```

5. Crie uma classe “Alimento” que possua os atributos “descricao” (texto), “preco” (decimal) e “validade” (inteiro), crie dois métodos construtores e os métodos assessores para estes atributos. Crie também um método “VerificaValidade()” que imprimirá a validade dos alimentos (em dias) e um método “ImprimeInformacoes()” para imprimir a descrição concatenada com o preço e a validade. Crie duas classes filhas de “Alimento”, que serão “AlimentoPerecivel” com o atributo “tipo” (texto) e “AlimentoIndustrializado” com o atributo “fabricante” (texto). Crie dois métodos construtores e os métodos assessores para ambas as classes filhas. Considere a validade dos alimentos perecíveis como horas e dos alimentos industrializados como dias. Reescreva o método “VerificaValidade()” nas classes filhas imprimindo a validade em dias (classe “AlimentoPerecivel” ou em horas (AlimentoIndustrializado), dependendo o tipo do produto. Por fim reescreva o método “ImprimeInformacoes()” para imprimir a descrição concatenada com o preço, a validade e o tipo para a subclasse “AlimentoPerecivel” e para imprimir descrição concatenada com o preço, a validade e o fabricante para a subclasse “AlimentoIndustrializado”. Não esqueça de informar se a validade é em horas ou em dias.
   Crie uma classe “Teste” contendo o método “Main” que irá simular a compra de conjunto de alimentos, independentemente do tipo de alimento. Cada produto comprado deverá ser armazenado em um vetor na classe “Main”. Esse vetor/arraylist deve se chamar “carrinho” que simula o carrinho de compras de produtos alimentícios de um cliente em um e-commerce. Insira nesse “carrinho” vários alimentos perecíveis e alimentos industrializados e depois chame o método

"getDescrição" e "getValidade" de todos os objetos presentes no vetor/arraylist para o usuário do carrinho saber as informações dos produtos em seu carrinho.

6. Crie uma classe Conta, que possua um número e um saldo e os métodos para pegar saldo, depositar e sacar. Adicione o atributo número da conta e saldo. Crie os métodos assessores, Deposita(double) e Saca(double). Crie também um método chamado Atualiza para atualizar essa conta de acordo com uma taxa percentual fornecida.
   Crie duas subclasses da classe Conta: ContaCorrente e ContaPoupanca. Ambas terão o método Atualiza reescrito: A ContaCorrente deve atualizar-se com o dobro da taxa e a ContaPoupanca deve atualizar-se com o triplo da taxa.
   Além disso, a ContaCorrente deve reescrever o método Deposita, a fim de retirar uma taxa bancária de dez centavos de cada depósito.
   Por fim, crie uma classe com método "*Main*" e instancie essas classes implementadas, faça um depósito em cada uma das instâncias, atualize-as e veja o resultado (saldo).
   Por fim, crie uma classe "Relatorio" que será responsável por mostrar a movimentação de todas as contas bancárias. Essa classe terá um método chamado "MostraMovimentacao" que receberá como parâmetro uma conta qualquer e imprimirá o número da conta movimentada e o saldo atual dessa conta.

   DESAFIO: Crie uma classe chamada "*Movimentacao*" que seja responsável por fazer a atualização de todas as contas bancárias, ou seja, todas as movimentações (saldo) das contas serão realizadas nessa classe.
   Além disso, conforme atualiza o valor (saldo) nas contas, o banco quer saber o montante total de dinheiro até o momento. Por isso, é necessário ir guardando o "saldoTotal" e adicionar um *Getter* à classe para pegar esse saldo.
   Essa classe terá um método chamado "AtualizaValorConta", que receberá como parâmetro uma conta qualquer, o tipo de movimentação e o valor a ser movimentado.
   A partir desse método serão realizados os movimentos de saque, deposito e atualização das contas. Esse método deverá gerar um relatório com o saldo anterior e saldo novo da conta que está sendo realizada a movimentação e o montante total de dinheiro do banco (saldoTotal).