

 <p>INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA PIAUI</p>	<p><b>INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ</b></p> <p><b>Curso: ADS</b></p> <p><b>Disciplina: Programação Orientada a Objetos</b></p> <p><b>Professor: Ely</b></p>
--	--

### Exercício 07

1. As classes **Carro**, **Veiculo** e **CarroEletrico** são bem semelhantes. Reescreva as classes usando herança para que os atributos duplicados não sejam mais necessários.

<pre>class Veiculo {     placa: String;     ano: number; }</pre>	<pre>class Carro {     placa: String;     ano: number;     modelo: String; }</pre>
<pre>class CarroEletrico {     placa: String;     ano: number;     modelo: String;     autonomiaBateria: number; }</pre>	

2. Crie uma classe Calculadora com:
  - a. Dois atributos privados chamados representando dois operandos;
  - b. Crie um construtor que inicializa os atributos;
  - c. Crie um método que retorna a soma dos dois atributos;
  - d. Teste a classe.
3. Crie uma classe chamada CalculadoraCientifica que herda da classe Calculadora do exercício passado e:
  - a. Implemente um método chamado exponenciar que retorne o primeiro operando elevado ao segundo;
  - b. Teste a classe;
  - c. Foi necessária alguma modificação em Calculadora para o acesso aos atributos?
4. Considerando a implementação da aplicação bancária, implemente:
  - a. Implemente na classe Banco o método renderJuros(numero: string): void, onde:
    - i. É passado por parâmetro o número de uma poupança e feita uma consulta para ver se a conta existe. Note que a consulta não se altera sendo Conta ou Poupança;
    - ii. Caso a poupança seja encontrada, teste se realmente se trata de uma poupança com o operador instanceof, desconsidere a operação caso contrário;

- iii. Caso seja, faça um cast e invoque o método renderJuros da própria instância encontrada;
  - iv. Teste o método da classe Banco passando tanto um número de poupança como de conta passados inseridos anteriormente;
  - v. Altere a aplicação anteriormente sugerida para ter a opção de menu "Render Juros".
- b. Adicione a aplicação para também permitir o cadastro da ContaImposto feita em sala de aula;
  - c. Incremente a implementação da aplicação para recuperar de um arquivo texto para o array contas salvas em um arquivo contas.txt com um formato semelhante ao abaixo:  
111-1; 40; C  
222-2; 10.65, CP; 0.5  
333-3; 2.00; CI; 0.38  
444-4; 140; CP; 0.5

Onde os campos separados por ponto-e-vírgula são o número, o saldo, o tipo da conta e, no caso de conta imposto e conta poupança, a taxa de desconto e taxa de juros.

Pesquise uma biblioteca de leitura e escrita de arquivos e deixe essa e a próxima opção disponíveis para o usuário escolher

- d. Implemente também uma funcionalidade de gravar no mesmo arquivo o conteúdo do array de contas
5. Suponha um sistema de controle de estoque de produtos e implemente:
- a. Duas classes: Produto e ProdutoPerecível;
  - b. A classe Produto tem atributos privados representando identificador, descrição, quantidade de produtos em estoque e valor unitário;
  - c. ProdutoPerecível tem as mesmas características de Produto, porém possui a mais um atributo representando a data da validade (<https://www.javatpoint.com/typescript-date-object>). Use herança;
  - d. Produto possui dois métodos para repor e dar baixa. A e ambos somam e subtraem respectivamente uma quantidade passada por parâmetro do atributo quantidade;
  - e. Um produto perecível possui um método que diz se um produto está válido ou não comparando sua data de validade com a data atual;
  - f. Use sobrescrita, ou seja, reescreva os métodos repor e dar baixa para que não seja possível executar a ação caso o produto não esteja na validade;
  - g. Crie uma classe chamada Estoque que possui um atributo privado representando um array de produtos (Produto ou ProdutoPerecível);
  - h. Implemente métodos para inserir, consultar pelo atributo id, excluir, repor e dar baixa nos produtos na classe estoque;
  - i. Crie validações para não deixar serem incluídos produtos com mesmo id ou mesma descrição. Para isso, crie uma consulta adicional através de um

método `existe(id: number, descricao: string): boolean` e use-o no método `incluir`.

- j. Os métodos `repor` e `dar baixa` na classe `estoque` chamam os métodos da classe `produto` finalmente alterar a quantidade;
- k. Os vários métodos da classe devem levar em conta se o produto existe, para isso, use o método `existe` ou `consultar`;
- l. Implemente um método que liste todos os produtos perecíveis vencidos.