

	INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ Curso: ADS Disciplina: Programação Orientada a Objetos Professor: Ely
--	---

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:
- Dois tributos privados chamados representando dois operandos;
 - Crie um construtor que inicializa os atributos;
 - Crie um método que retorna a soma dos dois atributos;
 - 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?

R: Sim, foi necessário adicionar getters dentro da classe Calculadora para conseguir utilizar o `_operando1` e o `_operando2` dentro da classe CalculadoraCientifica.

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 a possibilidade de ter o cadastro de Contalimposto feita em sala de aula. Foi necessário alterar alguma coisa na classe Banco ou apenas na classe App?

R: Foi preciso adicionar `Contalimposto` dentro do export da classe Banco e no import da classe APP além da mudança no método `inserirConta` da classe APP

para que também houvesse a possibilidade de inserir uma ContalImposto.

5. Dadas as três classes abaixo:

<pre>class Empregado { salario: number = 500; calcularSalario(): number { ...} }</pre>	<pre>class Diarista extends Empregado { calcularSalario(): number { ...} }</pre>
<pre>class Horista extends Diarista { calcularSalario(): number { ...} }</pre>	

Implemente os métodos calcularSalario() de cada classe da seguinte forma:

- Empregado: apenas retorna o valor do atributo salário;
- Diarista: sobrescreve calcularSalario, chamando o método homônimo de Empregado e dividindo o resultado por 30;
- Horista: sobrescreve calcularSalario, chamando o método homônimo de Diarista e dividindo o resultado por 24.

6. Crie uma classe Pessoa com:

- Atributos privados `_nome` (tipo string) e `_sobrenome` (tipo string). Cada um desses atributos deve ter métodos para lê-los (getters).
- Um método get chamado `nomeCompleto` que não possui parâmetros de entrada e que retorna a concatenação do atributo `nome` com o atributo `sobrenome`.
- Um construtor que recebe como parâmetros o nome e o sobrenome da pessoa e inicializa respectivamente os atributos `nome` e `sobrenome`.

7. Crie uma subclasse de Pessoa, chamada Funcionario que deve possuir:

- Os atributos privados `_matricula` do tipo string e `_salario` do tipo number, com seus respectivos métodos para leitura.
- O salário de um funcionário jamais poderá ser negativo. Todo funcionário recebe seu salário em duas parcelas, sendo 60% na primeira parcela e 40% na segunda parcela. Assim, escreva os métodos `calcularSalarioPrimeiraParcela` que retornam o valor da primeira parcela do salário (60%) e `calcularSalarioSegundaParcela` que retorna o valor da segunda parcela do salário (40%).

8. Uma subclasse de Funcionario, chamada Professor tendo:

- a. Um atributo `_titulacao` (string) com seu método de leitura;
 - b. Todo professor recebe seu salário em uma única parcela. Assim, deve-se sobrescrever os métodos `calcularSalarioPrimeiraParcela` e `calcularSalarioSegundaParcela`. O método `calcularSalarioPrimeiraParcela` da classe Professor deve retornar o valor integral do salário do professor e o método `calcularSalarioSegundaParcela` do professor deve retornar o valor zero.
9. Crie uma classe chamada Folha de pagamento que no construtor receba um array de Pessoa e inicialize um atributo do mesmo tipo. Crie um método chamado `calcularPagamentos()` que retorna um valor que represente o total de salários dos elementos do array. Note que você deve considerar o salário apenas de funcionários e professores.

Crie testes de todos os métodos das classes das questões anteriores.

R: Códigos de respostas dos métodos das classes presentes nos arquivos externos: `veiculo.ts`(questão 1), `calculadora.ts`(questão 2 e 3), `banco.ts` e `app.ts`(questão 4), `empregado.ts`(questão 5) e `pagamento.ts`(questão 6 a 9).