Desafio #1.1
Fundamentos do Javascript

Instruções

- Organize os códigos das questões em um único projeto de forma que, caso a mesma classe/função seja usada em mais de uma questão, não haja duplicidade de código.
- As questões devem ser desenvolvidas individualmente.
- O projeto deve ser versionado e disponibilizado no Github.

Questões

1. Programa **questao1.js**: crie a classe **Vertice** e implemente nessa classe:

   - Atributos numéricos **x** e **y** privados com leitura pública.
   - Construtor para inicializar os valores de **x** e **y**.
   - Método getter **distancia** para calcular a distância euclidiana de um vértice a outro.
   - Método **move** para mover o vértice para outra posição (x, y).
   - Método **equals** para verificar se dois vértices são iguais.

   Em seguida, leia valores do usuário para criar 3 vértices e chamar os métodos implementados na classe.

2. Programa **questao2.js**: usando a classe **Vertice** do exercício anterior, crie a classe **Triangulo**, que possui 3 vértices (privados com leitura pública). Nessa classe implemente:

   - Construtor para inicializar os vértices do triângulo. Gere uma exceção caso os vértices não formem um triângulo.
   - Método **equals** para verificar se dois triângulos são iguais.
   - Método getter **perimetro** para retornar o perímetro do triângulo.
   - Método **tipo** para retornar o tipo do triângulo (equilátero, isósceles ou escaleno).
   - Método **clone** para clonar um triângulo.
   - Método getter **area** para retornar a área do triângulo. Para calcular a área do triângulo use:

   onde a, b e c são os lados do triângulo e S é o perímetro dividido por 2, ou seja S = (a+b+c)/2.

Em seguida, leia valores do usuário para criar 3 triângulos e chamar os métodos implementados na classe.

3. Programa **questao3.js**: usando a classe **Vértice** do exercício anterior, crie a classe **Polígono**, que possui 3 ou mais vértices. Nessa classe implemente:

   - Construtor para inicializar os vértices do polígono (pelo menos 3 vértices). Gere uma exceção caso o polígono não tenha ao menos 3 vértices.
   - Método booleano **addVertice** para adicionar um novo vértice **v** ao polígono. Se o vértice já existe no polígono o método não deve adicioná-lo novamente e retornar falso.
   - Método getter **perimetro** para retornar o perímetro do polígono.
   - Método getter **qtdVertices** para retornar a quantidade de vértices do polígono.

   Em seguida, leia valores do usuário para criar um polígono e chamar os métodos implementados na classe.

4. Programa **questao4.js**: crie uma classe **Turma** que possui uma lista de **Alunos**. Cada aluno tem matrícula e nome (obrigatórios) e duas notas (P1 e P2) que inicialmente estão sem valor. Durante o semestre os alunos devem realizar essas provas, mas podem faltar a uma delas ou às duas. Crie métodos para:

   - Inserir um aluno na turma. Não podem ser inseridos dois alunos com a mesma matrícula.
   - Remover um aluno da turma a partir da sua matrícula.
   - Lançar a nota (seja ela P1 ou P2) de um aluno.
   - Imprimir os alunos da turma em ordem alfabética de acordo com o layout a seguir. A nota final é calculada como: (a) NF = (P1 + P2) / 2, para quem compareceu às duas provas; (b) NF = P1 / 2 ou NF = P2 / 2, para quem faltou a uma das provas, e; (c) NF = 0, para quem faltou às duas provas. Use uma casa decimal para as notas.

```
-----------------------------------------
Matricula Nome              P1   P2   NF
-----------------------------------------
  12345   Ana de Almeida   8.0  9.5  8.8
  23456   Bruno Carvalho   7.0    -  3.5
  34567   Fernanda Abreu     -  8.5  4.3
  45678   Joao Santos        -    -  0.0
-----------------------------------------
```

   Em seguida, leia dados dos alunos e suas notas e imprima a lista de alunos.

5. Programa **questao5.js**: crie uma aplicação que faz a entrada de dados pelo console dos dados de um cliente. Todos os dados deverão ser convertidos para os tipos adequados de acordo com as regras da tabela a seguir:

Campo	Regras	Tipo
Nome	Pelo menos 5 caracteres	string
CPF	Exatamente 11 dígitos	Number
Data de nascimento	Lida no formato DD/MM/AAAA O cliente deve ter pelo menos 18 anos na data atual	Date
Renda mensal	Valor Lida com duas casas decimais e vírgula decimal	Number
Estado civil	C, S, V ou D (maiúsculo ou minúsculo)	String
Dependentes	0 a 10	Number

Caso o dado fornecido não obedeça à regra, o programa deve emitir a mensagem de erro adequada e solicitá-lo novamente. Ao final, os dados corretos deverão ser impressos na tela: CPF com a máscara 999.999.999-99, renda com 2 casas decimais e data com a máscara dd/mm/aaaa.