Exercícios de Revisão - Parte 1
Unidade 1 - Linguagem C#

Resolva todos os exercícios de revisão dentro de uma mesma solução (*solution*) no Visual Studio e crie um projeto para cada um deles. Caso a mesma classe seja usada em mais de um exercício, não duplique o código e use referências.

1. Crie a classe **Piramide** e implemente nessa classe:
   - Propriedade inteira **N** (N ≥ 1).
   - Construtor para inicializar o valor de **N**. Gere uma exceção caso N < 1.
   - Método **Desenha** que imprime uma pirâmide com os números de 1 a N. Por exemplo, se N = 4, deverá ser desenha a seguinte pirâmide:

```
   1
  121
 12321
1234321
```

2. Crie a classe **Vertice** e implemente nessa classe:
   - Propriedades reais **X** e **Y** (leitura pública e escrita privada)
   - Construtor para inicializar os valores de **X** e **Y**.
   - Método **Distancia** para calcular a distância euclidiana de um vértice a outro.
   - Método **Move** para mover o vértice para outra posição (x, y).
   - Método para verificar se dois vértices são iguais.

3. Usando a classe **Vértice** do exercício anterior, crie a classe **Triangulo**, que possui 3 vértices (leitura pública e escrita privada). Nessa classe implemente:
   - Construtor para inicializar os vértices do triângulo. Gere uma exceção caso os vértices não formem um triângulo.
   - Método para verificar se dois triângulos são iguais.
   - Propriedade **Perimetro** para retornar o perímetro do triângulo.
   - Propriedade **Tipo** para retornar o tipo do triângulo (equilátero, isósceles ou escaleno). Dica: use um tipo enumerado.
   - Propriedade **Area** para retornar a área do triângulo. Para calcular a área do triângulo use:

$$área = \sqrt{S \,.\, (S - a).\, (S - b).\, (S - c)}$$

   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.

4. Usando a classe **Vértice** do exercício anterior, crie a classe **Poligono**, 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 **RemoveVertice** para remover um vértice **v** do polígono. Gerar exceção caso o polígono fique com menos de 3 vértices.
   - Método para retornar o perímetro do polígono.
   - Propriedade para retornar a quantidade de vértices do polígono.

5. Crie a classe **Intervalo** para representar um intervalo de tempo com data/hora inicial e final. Dica: use Datetime. Nessa classe implemente:
   - Construtor para inicializar a data/hora inicial e final. Gere uma exceção caso data/hora inicial > data/hora final.
   - Data/hora inicial e final não podem ser alterados.
   - Método booleano **TemIntersecao** que verifica se um intervalo tem interseção com outro intervalo ou não.
   - Método para verificar se dois intervalos são iguais.
   - Propriedade **Duracao** para retornar a duração de um intervalo. Dica: use TimeSpan.

6. Usando a classe **Intervalo** do exercício anterior crie a classe **ListaIntervalo** que implementa uma lista de intervalos. Nessa classe implemente:
   - Método **Add** para adicionar um novo intervalo à lista de forma que não seja possível adicionar um novo intervalo **i** à lista caso haja uma interseção de **i** com algum elemento da lista.
   - Método **Imprime** para imprimir a lista ordenada por data/hora inicial.

7. Crie uma aplicação que faz a entrada de dados pelo console dos dados de um cliente. Todos os dados deverão ser lidos em formato **string** e 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	long
Data de nascimento	Lida no formato DD/MM/AAAA O cliente deve ter pelo menos 18 anos na data atual	DateTime
Renda mensal	Lida com duas casas decimais e vírgula decimal	float
Estado civil	C, S, V ou D (maiúsculo ou minúsculo)	char
Dependentes	0 a 10	int

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.