

LISTA II

EXERCÍCIO 01

a) Crie uma classe denominada Ponto.cs que irá representar um ponto no plano cartesiano (x, y) . Essa classe conterá apenas dois atributos reais denominados X e Y . Cada atributo pode assumir qualquer valor real.

b) Além disso, essa classe conterá os seguintes métodos listados abaixo:

- `public Ponto(double x, double y)`
- `public static double Distancia(Ponto p1, Ponto p2)`
- `public static List<double> Reta(Ponto p1, Ponto p2)`

O primeiro método trata-se de um construtor utilizado para inicializar os atributos.

O segundo método é usado para calcular a distância entre dois pontos passados como parâmetro. A distância entre um ponto $p1$ e um ponto $p2$ no plano cartesiano é dada pela fórmula abaixo:

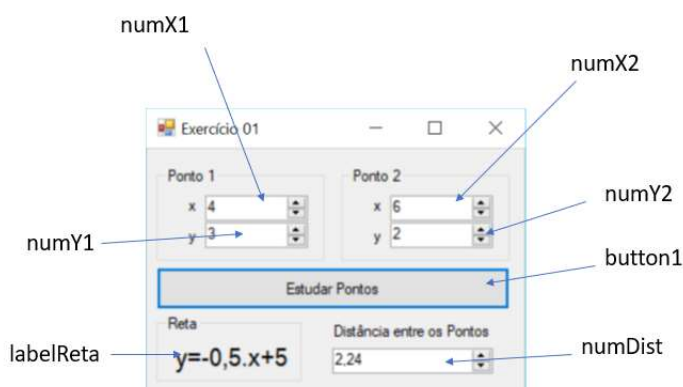
$$Dist(p1, p2) = \sqrt{(x_{p1} - x_{p2})^2 + (y_{p1} - y_{p2})^2}$$

O terceiro método retorna uma lista contendo apenas dois valores: o coeficiente angular e o coeficiente linear da reta que passa pelos dois pontos $p1$ e $p2$. Em outras palavras, na primeira posição da lista nós temos o valor do coeficiente angular e na segunda posição da lista nós temos o valor do coeficiente linear.

c) Crie um aplicativo de WindowsForms conforme ilustrado ao lado e implemente o método

- `private void button1_Click(object sender, EventArgs e).`

Este método deverá preencher o conteúdo dos objetos **labelReta** e **numDist** de acordo com os valores passados como entrada.



EXERCÍCIO 02

De acordo com os conceitos de Movimento Uniformemente Variado (MUV), implemente uma classe denominada Carro que se desloca de acordo com esse tipo de movimentação.

- Implemente uma classe Carro. Como sabemos um Movimento Uniforme Variado é definido por uma posição, uma velocidade e uma aceleração. Insira esses três atributos S, V e A na classe Carro. Insira também um atributo Piloto, contendo o nome do motorista do carro.
- Determine se há necessidade de aplicar o encapsulamento em algum ou todos os atributos da classe. Se houver a necessidade, crie os métodos Getters e Setters para o(s) atributo(s) e justifique com um comentário no código.
- Implemente um construtor para esta classe. Esse construtor irá receber quatro valores reais (s, v, a e nome) como parâmetros e irá atribuí-los aos atributos da classe.
- Implemente o método `public void Mover(double t)` que irá atualizar a posição e a velocidade do objeto Carro sempre que for chamado.
- No método Main da classe Programa.cs, crie duas instâncias de carro. A primeira se encontra na posição 5m, está parada e seu motor fornece uma aceleração de 10m/s^2 - o nome do motorista é "Pedro". A segunda instância se encontra na posição 20m, também está parada e apresenta aceleração de 20m/s^2 - a motorista se chama "Maria".
- Imagine agora que há uma competição entre esses dois carros. Continue a implementação do método anterior movendo os dois carros a incrementos de 1s (em relação à variável tempo). Seu programa deverá indicar em que instante de tempo "Maria" ultrapassou "Pedro".

EXERCÍCIO 03

- Crie um programa do Console que seja capaz de ler valores inteiros enquanto o usuário digitar "sim". Esses valores deverão ser armazenados em um vetor de inteiros ou uma lista de inteiros (você decide).
- Dizemos que uma sequência de inteiros positivos é **k-alternante** se for composta alternadamente por segmentos de números pares de tamanho k e segmentos de números ímpares de tamanho k.

Exemplos:

A sequência: 1 3 6 8 9 11 2 4 7 6 8 é 2-alternante.

A sequência: 2 1 4 7 8 9 12 é 1-alternante.

A sequência: 4 2 3 1 6 4 2 9 3 não é alternante.

Sabendo disso, faça o que é pedido:

- Escreva um método com a assinatura
 - `public static int kAlternante(int[] valores)` ou
 - `public static int kAlternante(List<int> valores);`

Esse método deverá retornar um inteiro indicando se o vetor ou lista passada como parâmetro é uma sequência k-alternante. Se retornar 0, não é uma sequência k-alternante, se retornar um inteiro x então é uma sequência x-alternante.

- Chame o método kAlternante no método Main para testar se os valores que você leu tratam-se de uma sequência k-alternante.