Summary

Information

Class:	SuperCodigo.Operacoes
Assembly:	SuperCodigo
File(s):	D:\SuperCodigo\SuperCodigo\Operacoes.cs

Line coverage

100%

Covered lines:	50
Uncovered lines:	0
Coverable lines:	50
Total lines:	82
Line coverage:	100%

Branch coverage

100%

Covered branches:	36
Total branches:	36
Branch coverage:	100%

Method coverage

Method coverage is only available for sponsors.

Upgrade to PRO version

Metrics

Method	Branch coverage	Cyclomatic complexity	Line coverage
ObterPosicaoCaractere(...)	100%	10	100%
ObterElementoFibonnaci(...)	100%	6	100%
DeterminarTipoTriangulo(...)	100%	20	100%

File(s)

D:\SuperCodigo\SuperCodigo\Operacoes.cs

#	Line	Line coverage
	1	using System;
	2	
	3	namespace SuperCodigo
	4	{
	5	public class ConstantesOperacoes
	6	{
	7	public const string CADEIA_CARACTERES_CADEIA_INVALIDA = "Cadeia Inválida.";
	8	public const string CADEIA_CARACTERES_CARACTERE_INVALIDO = "Caractere inválido.";
	9	public const string FIBONNACI_MAIOR_QUE_ZERO = "n deve ser maior que zero.";
	10	}
	11	
	12	public class Operacoes
	13	{
	14	public int ObterPosicaoCaractere(string cadeia, string caractere)
4	15	{
4	16	P if (cadeia.Length == 0 || cadeia.Length > 20)
1	17	throw new ArgumentException(ConstantesOperacoes.CADEIA_CARACTERES_CADEIA_INVALIDA);
	18	
3	19	P if (caractere.Length != 1)
1	20	throw new ArgumentException(ConstantesOperacoes.CADEIA_CARACTERES_CARACTERE_INVALIDO);
	21	
16	22	P for (int i = 1; i <= cadeia.Length; i++) // pelo enunciado do problema a primeira posição é 1 e não 0
7	23	{
7	24	P if (cadeia[i-1].ToString() == caractere)
1	25	return i;
6	26	}
	27	
1	28	return -1;
2	29	}
	30	
	31	public int ObterElementoFibonnaci(int n)
3	32	{
3	33	P if (n < 1)
1	34	throw new ArgumentException(ConstantesOperacoes.FIBONNACI_MAIOR_QUE_ZERO);
	35	
6	36	int elementoAnterior1 = 1, elementoAnterior2 = 1, elementoAtual = elementoAnterior1;
	37	
2	38	P if (n == 2)
1	39	elementoAtual = elementoAnterior1;
	40	else
1	41	{
12	42	P for (int i = 3; i <= n; i++)
5	43	{
5	44	elementoAtual = elementoAnterior1 + elementoAnterior2;
5	45	elementoAnterior1 = elementoAnterior2;
5	46	elementoAnterior2 = elementoAtual;
5	47	}
1	48	}
	49	
2	50	return elementoAtual;
2	51	}
	52	
	53	public string DeterminarTipoTriangulo(int a, int b, int c)
6	54	{
6	55	string tipo = "ESCALENO";
	56	
6	57	P if (a <= 0 || b <= 0 || c <= 0)
1	58	tipo = "INEXISTENTE";
	59	else
5	60	{
5	61	P if (!(a + b > c && a + c > b && b + c > a))
1	62	tipo = "INEXISTENTE";
	63	else
4	64	{
4	65	P if (a == b)
2	66	{
2	67	tipo = "ISOSCELES";
2	68	P if (b == c)
1	69	tipo = "EQUILATERO";
2	70	}
	71	else
2	72	{
2	73	P if (b == c || a == c)
1	74	tipo = "ISOSCELES";
2	75	}
4	76	}
5	77	}
	78	
6	79	return tipo;
6	80	}
	81	}
	82	}

Methods/Properties

- ObterPosicaoCaractere(System.String,S...
- ObterElementoFibonnaci(System.Int32)
- DeterminarTipoTriangulo(System.Int32,S...