|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nº | Valor introducido | Resultado esperado | Resultado Obtenido | Resultado de la prueba |
| NumeroEsPrimo3 | 3 | “Si es primo” | “Si es primo” | OK |
| NumeroEsPrimo4 | 4 | “No es primo” | “No es primo” | OK |
| NumeroEsPrimo0 | 0 | Error encontrado | Es primo | Failed |
| NumeroEsPrimoMinValue | MinValue | “Si es primo” | Error al intentar dividir por 0 | Failed |
| NumeroEsPrimoMaxValue | MaxValue | “Si es primo” | Error al intentar dividir por 0 | OK |
| NumeroEsPrimoMinValue2 | MinValue -1 | “No es primo” | “No es primo” | OK |
| NumeroEsPrimoMaxValue2 | MaxValue -1 | “No es primo” | “No es primo” | OK |

CASO DE PRUEBA 1 : PruebaApp1

Para realizar estos casos de prueba, he enlazado la clase en donde esta situada la prueba, después he creado un resultado esperado, y le he ido pasando valores introducidos, en este caso, he probado con la clase numeros pimos, la cual solo me ha producido 2 errores

CASO DE PRUEBA 2 : PruebaApp2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Valor introducido | Resultado esperado | Resultado Obtenido | Resultado de la prueba |
| NumeroEsPrimo3 | 3 | “Si es primo” | “Si es primo” | OK |
| NumeroEsPrimo4 | 4 | “No es primo” | “No es primo” | OK |
| NumeroEsPrimo0 | 0 | Error encontrado | Es primo | Failed |
| NumeroEsPrimoMinValue | MinValue | “Si es primo” | Error al intentar dividir por 0 | Failed |
| NumeroEsPrimoMaxValue | MaxValue | “Si es primo” | Error al intentar dividir por 0 | OK |
| NumeroEsPrimoMinValue2 | MinValue -1 | “No es primo” | “No es primo” | OK |
| NumeroEsPrimoMaxValue2 | MaxValue -1 | “No es primo” | “No es primo” | OK |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Valor introducido | Resultado esperado | Resultado Obtenido | Resultado de la prueba |
| Cadena1 | 1 | 1,2,3,4,5,6,7,8,9,10,11  ,12,13,14,15,\n | 1,2,3,4,5,6,7,8,9,10,11,  12,13,14,15,\n | OK |
| Valor0 | 0 | “Numero no valido” | “Numero no valido” | OK |
| Cadena1 | 2 | 1,2,3,4,5,6,7,8,9,10,11  ,12,13,14,15,\n1,2,3,4,5,6,  7,8,9,10,11  ,12,13,14,15,\n | 1,2,3,4,5,6,7,8,9,10,11  ,12,13,14,15,\n1,2,3,4,5,6,  7,8,9,10,11  ,12,13,14,15,\n | OK |
| CadenaNegativo | -1 | “Numero no valido” | “Numero no valido” | OK |

Este caso de prueba me ha llevado mucho tiempo, ya que tenía que respetar todos los espacios depurandolo varias veces, pero al final es esto, se expera una cadena de numeros respetando un valor que introduce el usuario, la clase es llamada desde la prueba unitaria y se realiza esto, con 0 y -1 nos da un fallo, lo cual es lo que buscamos , y solo aceptaría numeros positivos

CASO DE PRUEBA 3 : PruebaApp3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Valor introducido | Resultado esperado | Resultado Obtenido | Resultado de la prueba |
| base2exponente2 | basee = 2; exponente = 2; | 4 | 4 | OK |
| base2exponente0 | basee = 2; exponente = 0; | 1 | 1 | OK |
| base0exponente0 | basee = 0; exponente = 0; | 1 | 1 | OK |
| negativos | basee = -2;  exponente=-2; | “Introduzca un numero valido” | “Introduzca un numero valido” | OK |

Pasamos una base y un exponente desde la clase hasta la zona de pruebas unitarias correspondiente, alli efectuamos las pruebas que se ven, negando el acto de querer meter un numero negativo y cumpliendo perfectamente con las demas tareas, sin errores

LA APLICACIÓN 4 NO TIENE TIPO DE PRUEBAS PORQUE SE TRATA DE UN BOTON Y NO SE LE PUEDE HACER NINGUNA INSERCCION DE VALORES