

Reporte de Revisiones y Pruebas de Aceptación del programa de computadora (PRP-1).

Materia:

Modelado Orientado a Objetos.

Proyecto:

Marco con funciones de cálculo para operaciones estadísticas.

Fecha: junio 2022

Alumno:

Edgar Valentin Ruiz Padilla

Maestría en Ciencias Computacionales. Ingeniería de Software

Revisión y Pruebas de Aceptación del programa.

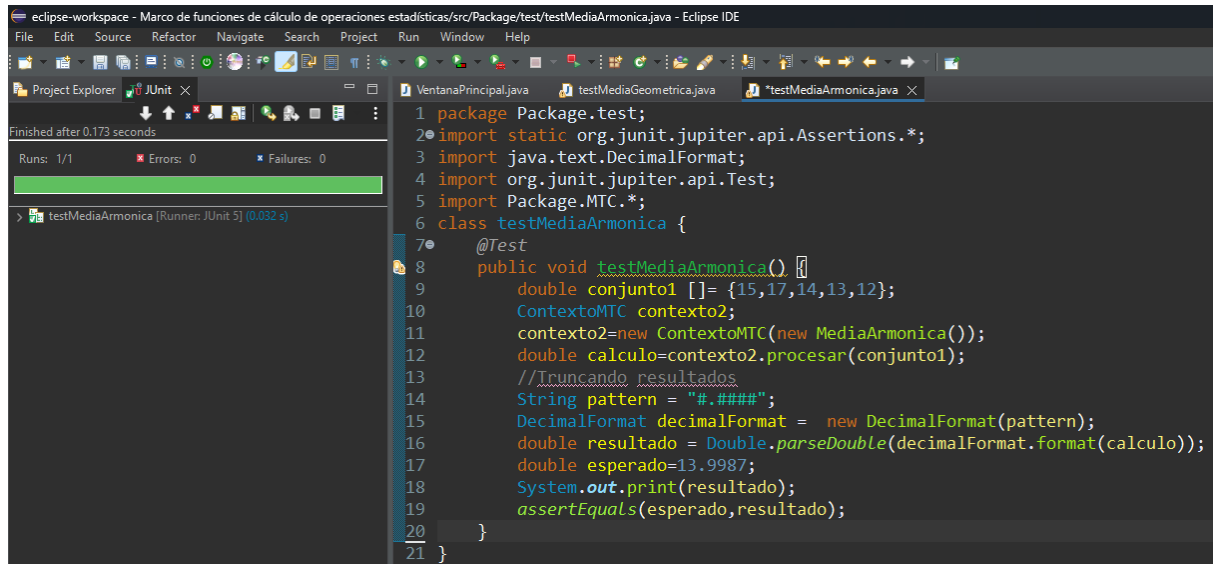
Características a probar	Pruebas unitarias
CU1: Calcular Media Aritmética	<p>Valor esperado:</p> $x_{avg} = \frac{21.3 + 38.4 + 12.7 + 41.6}{4} = 28.5$ 
CU2: Calcular Media Geométrica	<p>Valor esperado: La media geométrica de los números 2, 3 y 14 es igual a 4.37952</p> 
CU3: Calcular Media Armónica	<p>Valor esperado:</p>

Maestría en Ciencias Computacionales. Ingeniería de Software

$$H = \frac{5}{\frac{1}{15} + \frac{1}{17} + \frac{1}{14} + \frac{1}{13} + \frac{1}{12}}$$

$$H = \frac{5}{0,0667 + 0,0588 + 0,0714 + 0,0769 + 0,0833}$$

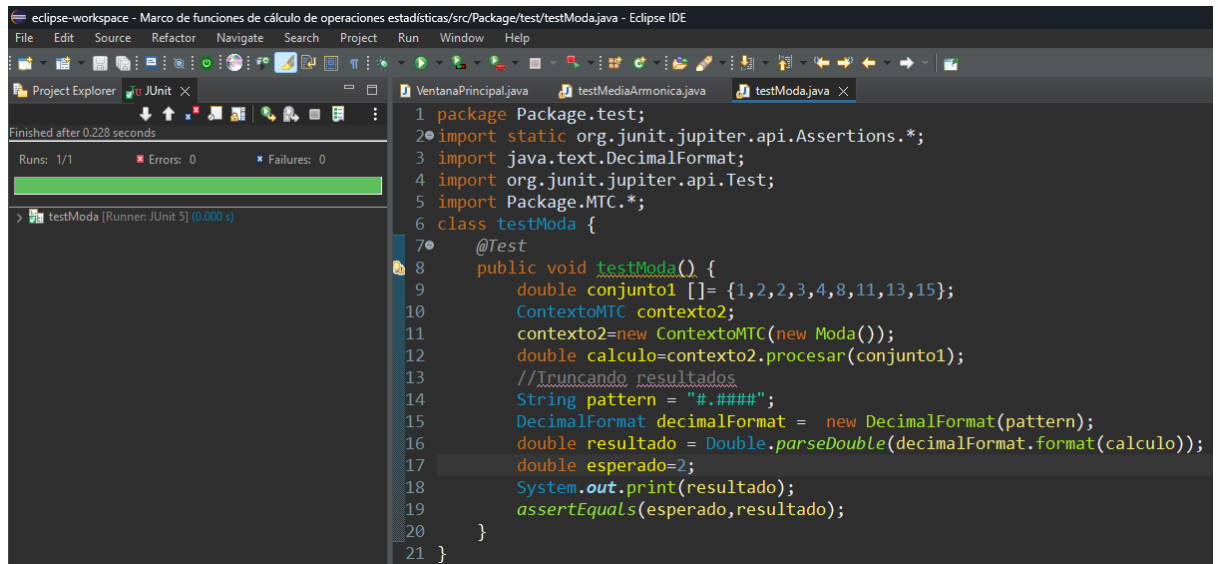
$$H = \frac{5}{0,3572} = 13,9987$$



```

1 package Package.test;
2 import static org.junit.jupiter.api.Assertions.*;
3 import java.text.DecimalFormat;
4 import org.junit.jupiter.api.Test;
5 import Package.MTC.*;
6 class testMediaArmonica {
7     @Test
8     public void testMediaArmonica() {
9         double conjunto1 []= {15,17,14,13,12};
10        ContextoMTC contexto2;
11        contexto2=new ContextoMTC(new MediaArmonica());
12        double calculo=contexto2.procesar(conjunto1);
13        //Truncando resultados
14        String pattern = "#.####";
15        DecimalFormat decimalFormat = new DecimalFormat(pattern);
16        double resultado = Double.parseDouble(decimalFormat.format(calculo));
17        double esperado=13.9987;
18        System.out.print(resultado);
19        assertEquals(esperado,resultado);
20    }
21 }
  
```

Valor esperado: la moda de los siguientes números 1,2,2,3,4,8,11,13,15 se obtiene que la moda es 2



```

1 package Package.test;
2 import static org.junit.jupiter.api.Assertions.*;
3 import java.text.DecimalFormat;
4 import org.junit.jupiter.api.Test;
5 import Package.MTC.*;
6 class testModa {
7     @Test
8     public void testModa() {
9         double conjunto1 []= {1,2,2,3,4,8,11,13,15};
10        ContextoMTC contexto2;
11        contexto2=new ContextoMTC(new Moda());
12        double calculo=contexto2.procesar(conjunto1);
13        //Truncando resultados
14        String pattern = "#.####";
15        DecimalFormat decimalFormat = new DecimalFormat(pattern);
16        double resultado = Double.parseDouble(decimalFormat.format(calculo));
17        double esperado=2;
18        System.out.print(resultado);
19        assertEquals(esperado,resultado);
20    }
21 }
  
```

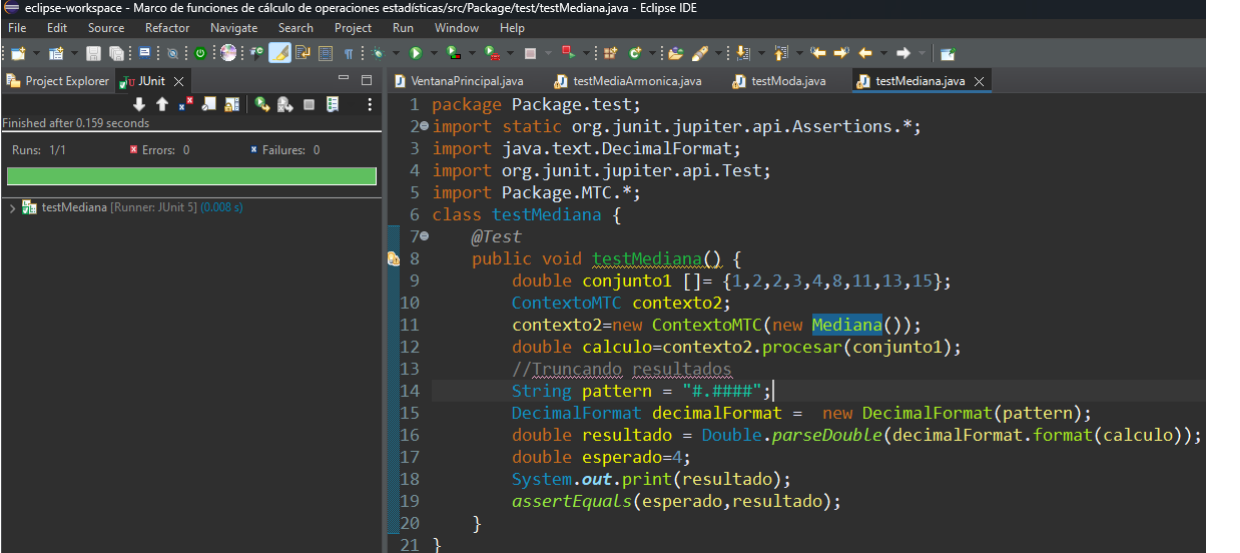
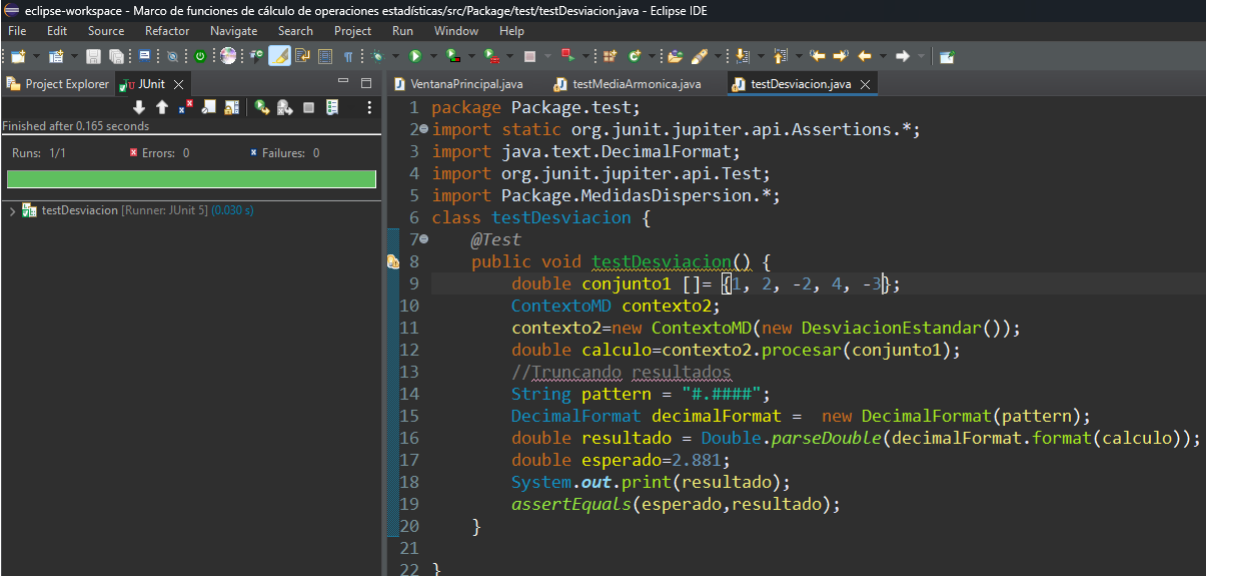
**CU4:
Calcular
Moda**

**CU5:
Calcular
Mediana**

Valor esperado: para el conjunto: 1,2,2,3,4,8,11,13,15
Se debe obtener 4 como valor de mediana.

Maestría en Ciencias Computacionales.

Ingeniería de Software

	 <pre> 1 package Package.test; 2 import static org.junit.jupiter.api.Assertions.*; 3 import java.text.DecimalFormat; 4 import org.junit.jupiter.api.Test; 5 import Package.MTC.*; 6 class testMediana { 7 @Test 8 public void testMediana() { 9 double conjunto1 []= {1,2,2,3,4,8,11,13,15}; 10 ContextoMTC contexto2; 11 contexto2=new ContextoMTC(new Mediana()); 12 double calculo=contexto2.procesar(conjunto1); 13 //Truncando resultados 14 String pattern = "#.###"; 15 DecimalFormat decimalFormat = new DecimalFormat(pattern); 16 double resultado = Double.parseDouble(decimalFormat.format(calculo)); 17 double esperado=4; 18 System.out.print(resultado); 19 assertEquals(esperado,resultado); 20 } 21 } </pre> <p>Finished after 0.159 seconds Runs: 1/1 Errors: 0 Failures: 0 testMediana [Runner: JUnit 5] (0.008 s)</p>
<p>CU6: Calcular Desviación Estándar</p>	<p>Valor esperado: Realizando el cálculo de la desviación estándar de los siguientes datos: 1, 2, -2, 4, -3 Se obtiene una desviación estándar de 2.88097, redondeando queda 2.881</p>  <pre> 1 package Package.test; 2 import static org.junit.jupiter.api.Assertions.*; 3 import java.text.DecimalFormat; 4 import org.junit.jupiter.api.Test; 5 import Package.MedidasDispersión.*; 6 class testDesviacion { 7 @Test 8 public void testDesviacion() { 9 double conjunto1 []= {1, 2, -2, 4, -3}; 10 ContextoMD contexto2; 11 contexto2=new ContextoMD(new DesviacionEstandar()); 12 double calculo=contexto2.procesar(conjunto1); 13 //Truncando resultados 14 String pattern = "#.###"; 15 DecimalFormat decimalFormat = new DecimalFormat(pattern); 16 double resultado = Double.parseDouble(decimalFormat.format(calculo)); 17 double esperado=2.881; 18 System.out.print(resultado); 19 assertEquals(esperado,resultado); 20 } 21 } </pre> <p>Finished after 0.165 seconds Runs: 1/1 Errors: 0 Failures: 0 testDesviacion [Runner: JUnit 5] (0.030 s)</p>
<p>CU7: Calcular Varianza</p>	<p>Valor esperado: Se realiza el cálculo de la varianza de los siguientes números: 21.3, 38.4, 12.7, 41.6 obteniendo como resultado del cálculo 190.36666</p>

Maestría en Ciencias Computacionales.

Ingeniería de Software

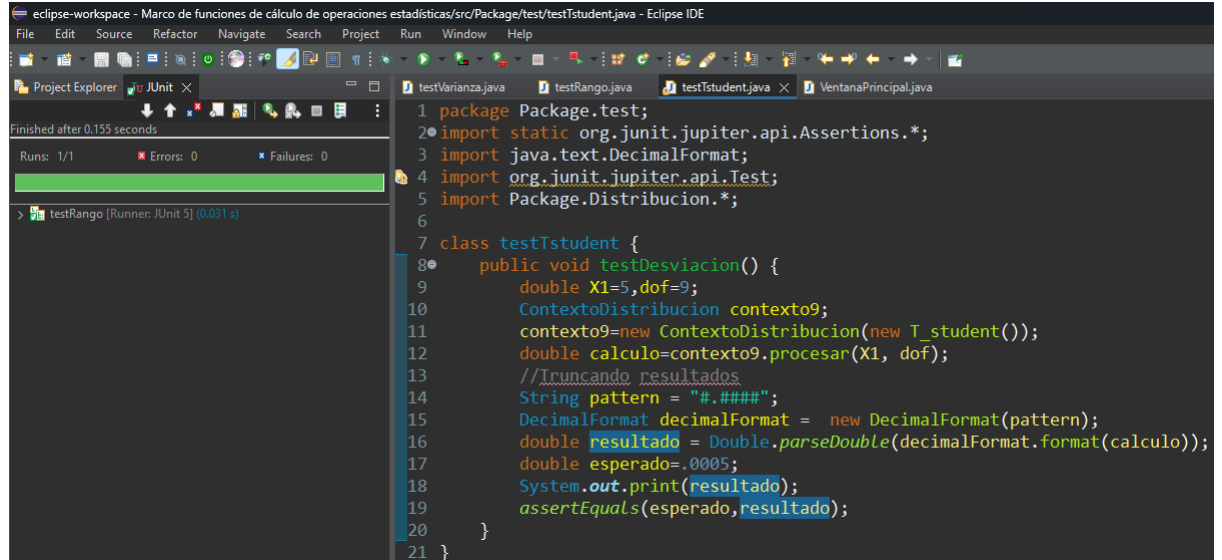
	 <pre> 1 package Package.test; 2 import static org.junit.jupiter.api.Assertions.*; 3 import java.text.DecimalFormat; 4 import org.junit.jupiter.api.Test; 5 import Package.MedidasDispersión.*; 6 7 class testVarianza { 8 @Test 9 public void testDesviación() { 10 double conjunto1 []= {21.3, 38.4, 12.7, 41.6 }; 11 ContextoMD contexto2; 12 contexto2=new ContextoMD(new Varianza()); 13 double calculo=contexto2.procesar(conjunto1); 14 //Truncando resultados 15 String pattern = "#.###"; 16 DecimalFormat decimalFormat = new DecimalFormat(pattern); 17 double resultado = Double.parseDouble(decimalFormat.format(calculo)); 18 double esperado=190.3667; 19 System.out.print(resultado); 20 assertEquals(esperado,resultado); 21 } 22 } </pre>
<p>CU8: Calcular Rango</p>	<p>Valor esperado: Al ingresar los siguientes datos: 1,2,2,3,4,8,11,13,15 se obtiene 14 de rango para el conjunto de números.</p>  <pre> 1 package Package.test; 2 import static org.junit.jupiter.api.Assertions.*; 3 import java.text.DecimalFormat; 4 import org.junit.jupiter.api.Test; 5 import Package.MedidasDispersión.*; 6 7 class testRango { 8 @Test 9 public void testDesviación() { 10 double conjunto1 []= {1,2,2,3,4,8,11,13,15}; 11 ContextoMD contexto2; 12 contexto2=new ContextoMD(new Rango()); 13 double calculo=contexto2.procesar(conjunto1); 14 //Truncando resultados 15 String pattern = "#.###"; 16 DecimalFormat decimalFormat = new DecimalFormat(pattern); 17 double resultado = Double.parseDouble(decimalFormat.format(calculo)); 18 double esperado=14; 19 System.out.print(resultado); 20 assertEquals(esperado,resultado); 21 } 22 } </pre>
<p>CU9: Calcular Correlación</p>	<p>Valor esperado: Se ingresara como primer conjunto de números: 186, 699, 132, 272, 291, 331, 199, 1890, 788, 1601 y como segundo conjunto de números: 15.0, 69.9, 6.5, 22.4, 28.4, 65.9, 19.4, 198.7, 38.8, 138.2 El valor calculado es: .9543157610641518</p>  <pre> 1 package Package.test; 2 import static org.junit.jupiter.api.Assertions.*; 3 4 class testCorrelacion { 5 @Test 6 public void testCorrelacion() { 7 double conjunto []= {186,699,132,272,291,331,199,1890,788,1601}; 8 double conjunto2 []= {15.0,69.9,6.5,22.4,28.4,65.9,19.4,198.7,38.8,138.2}; 9 AbsCorrelacion Corr=new Correlacion(); 10 double correlacion=Corr.calcular(conjunto,conjunto2); 11 //Truncando resultados 12 String pattern = "#.##"; 13 DecimalFormat decimalFormat = new DecimalFormat(pattern); 14 double resultado = Double.parseDouble(decimalFormat.format(correlacion)); 15 double esperado=0.95; 16 assertEquals(esperado,resultado); 17 } 18 } </pre>

Maestría en Ciencias Computacionales. Ingeniería de Software

CU11: Calcular Distribución T student

Valor esperado: Datos ingresados:
Se toma en cuenta (dof)=9
 $X_k=5$

El resultado debe ser: 5.042997581350438E -4



```

1 package Package.test;
2 import static org.junit.jupiter.api.Assertions.*;
3 import java.text.DecimalFormat;
4 import org.junit.jupiter.api.Test;
5 import Package.Distribucion.*;
6
7 class testTstudent {
8     public void testDesviacion() {
9         double X1=5,dof=9;
10        ContextoDistribucion contexto9;
11        contexto9=new ContextoDistribucion(new T_student());
12        double calculo=contexto9.procesar(X1, dof);
13        //Truncando resultados
14        String pattern = "#.####";
15        DecimalFormat decimalFormat = new DecimalFormat(pattern);
16        double resultado = Double.parseDouble(decimalFormat.format(calculo));
17        double esperado=.0005;
18        System.out.print(resultado);
19        assertEquals(esperado,resultado);
20    }
21 }
    
```

CU12: Calcular Regresión Multifactorial

Valor esperado:

Se considera calcular la regresión multifactorial de los datos de la siguiente tabla:

Se consideran
datos wk= 185
para una

	w	x	y	z
1	345	65	23	31.4
2	168	18	18	14.6
3	94	0	0	6.4
4	187	185	98	28.3
5	621	87	10	42.1
6	255	0	0	15.3

también estos
 $x_k= 150$ y $z_k=45$
predicción Z_k

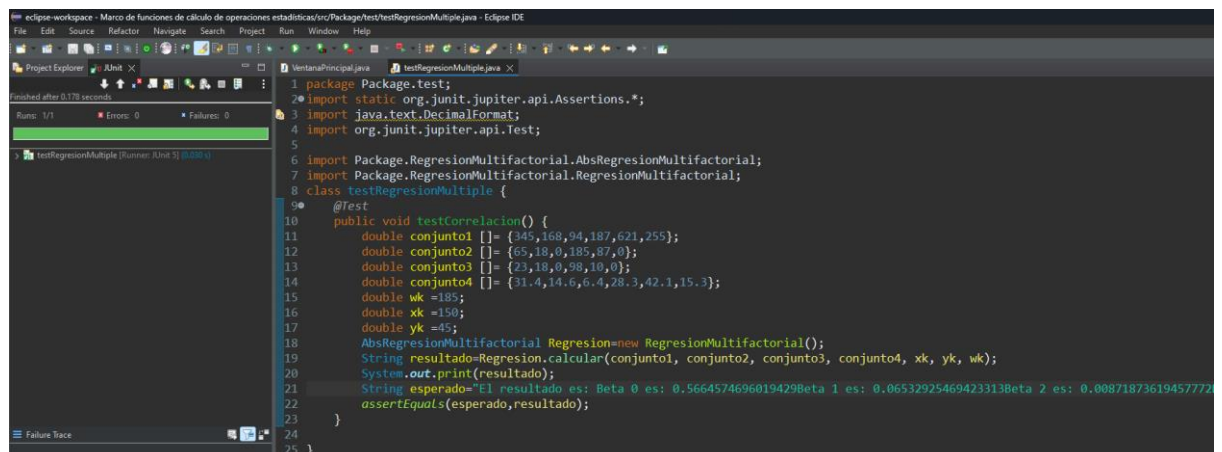
Beta
 $0=0.5664574696019$
Beta

$1=0.06532925469423$

Beta 2=0.00871873619457

Beta 3= 0.15104864761036

$Z_k= 20.757369$



```

1 package Package.test;
2 import static org.junit.jupiter.api.Assertions.*;
3 import java.text.DecimalFormat;
4 import org.junit.jupiter.api.Test;
5
6 import Package.RegresionMultifactorial.AbsRegresionMultifactorial;
7 import Package.RegresionMultifactorial.RegresionMultifactorial;
8 class testRegresionMultiple {
9     @Test
10    public void testCorrelacion() {
11        double conjunto1 []= {345,168,94,187,621,255};
12        double conjunto2 []= {65,18,0,185,87,0};
13        double conjunto3 []= {23,18,0,98,10,0};
14        double conjunto4 []= {31.4,14.6,6.4,28.3,42.1,15.3};
15        double wk =185;
16        double xk =150;
17        double yk =45;
18        AbsRegresionMultifactorial Regresion=new RegresionMultifactorial();
19        String resultado=Regresion.calcular(conjunto1, conjunto2, conjunto3, conjunto4, xk, yk, wk);
20        System.out.print(resultado);
21        String esperado="El resultado es: Beta 0 es: 0.5664574696019429Beta 1 es: 0.06532925469423313Beta 2 es: 0.00871873619457772";
22        assertEquals(esperado,resultado);
23    }
24 }
25 }
    
```

CU13: Calcular

Valor esperado:

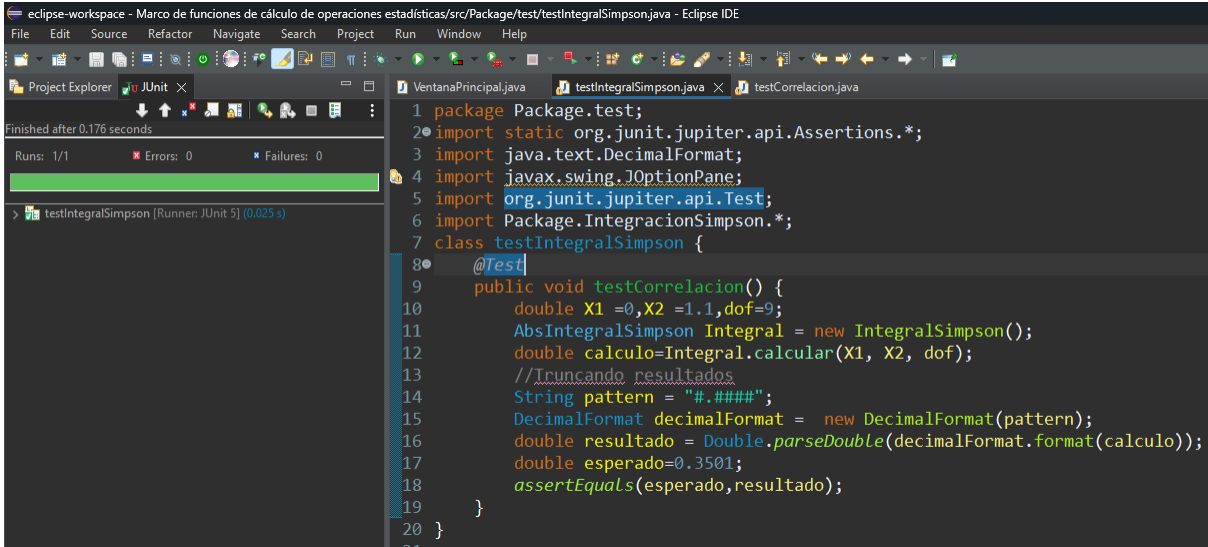
Intervalo de prueba	Integral Simpson
---------------------	------------------

Maestría en Ciencias Computacionales.

Ingeniería de Software

Integral Simpson

x	dof	p
0 to $x = 1.1$	9	0.35005864
0 to $x = 1.1812$	10	0.36757341



```

1 package Package.test;
2 import static org.junit.jupiter.api.Assertions.*;
3 import java.text.DecimalFormat;
4 import javax.swing.JOptionPane;
5 import org.junit.jupiter.api.Test;
6 import Package.IntegracionSimpson.*;
7 class testIntegralSimpson {
8     @Test
9     public void testCorrelacion() {
10         double X1 = 0, X2 = 1.1, dof = 9;
11         AbsIntegralSimpson Integral = new IntegralSimpson();
12         double calculo = Integral.calcular(X1, X2, dof);
13         //Truncando resultados
14         String pattern = "#.####";
15         DecimalFormat decimalFormat = new DecimalFormat(pattern);
16         double resultado = Double.parseDouble(decimalFormat.format(calculo));
17         double esperado = 0.3501;
18         assertEquals(esperado, resultado);
19     }
20 }

```