Hoja de trabajo 2 – Unit Test

Para la clase Calculadora se realizaron 5 pruebas distintas que comprobaran la funcionalidad de cada una de las implementaciones planeadas

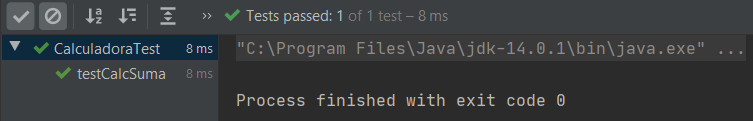
1. Test de suma

@Test  
public void testCalcSuma() {  
 *assertEquals*("3", calc.Calculo("2 1 +"));  
}

Para esta prueba se quiere comprobar el correcto funcionamiento de la implementación de suma a la calculadora

En esta prueba se suma 1 + 2, como resultado se espera un 3.

La prueba fue completada exitosamente.



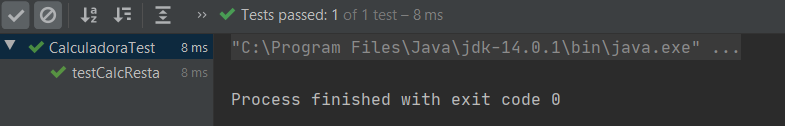
1. Test de resta

@Test  
public void testCalcResta(){  
 *assertEquals*("4", calc.Calculo("4 8 -"));  
}

Para esta prueba se quiere comprobar el correcto funcionamiento de la implementación de resta a la calculadora.

En esta prueba se resta 8 – 4, como resultado se espera un 4.

La prueba fue completada exitosamente



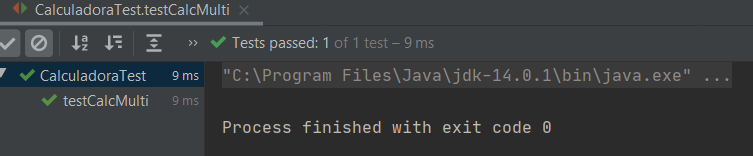
1. Test de multiplicación

@Test  
public void testCalcMulti() {  
 *assertEquals*("12", calc.Calculo("4 3 \*"));  
}

Para esta prueba se quiere comprobar la correcta implementación de la multiplicación en la calculadora

En esta prueba se multiplica 4 \* 3 y se espera un resultado de 12.

La prueba fue completada exitosamente



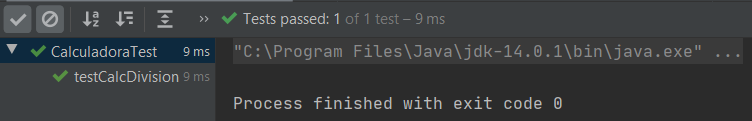
1. Test de división

@Test  
public void testCalcDivision(){  
 *assertEquals*("2", calc.Calculo("1 2 /"));  
}

Para esta prueba se quiere comprobar el correcto funcionamiento de la implementación de división en la calculadora.

En esta prueba se divide 2/1 y se espera un resultado de 2.

La prueba fue completada exitosamente.



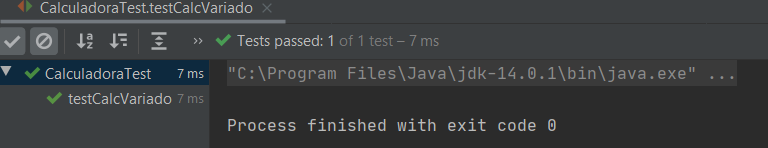
1. Test variado

@Test  
public void testCalcVariado() {  
 *assertEquals*("15", calc.Calculo("1 2 + 4 \* 3 +"));  
}

Para esta prueba se quiere comprobar el correcto funcionamiento de la implementación de varias operaciones dentro de la calculadora

En esta prueba se ingresa la operación (((1+2)\*4)+3) y se espera como resultado 15.

La prueba fue completada exitosamente



Para las pruebas unitarias de la clase StackVector se hicieron 4 pruebas para probar por completo su correcta funcionalidad.

1. Test de push y pop

@Test  
public void TestPushPop() {  
 stack.push("Prueba1");  
 *assertEquals*("Prueba1", stack.pop());  
}

Para este test se quiere probar la funcionalidad de almacenamiento del stack y como nos puede mostrar el contenido del ultimo objeto ingresado.

1. Test de peek

@Test  
public void TestPeek(){  
 stack.push("Touhou");  
 *assertEquals*("Touhou", stack.peek());  
}

Para este test se quiere probar la funcionalidad de que el stack devuelva el ultimo objeto ingresado.

1. Test de size

@Test  
public void Testsize() {  
 stack.push("CR7");  
 stack.push("Binario");  
 stack.push("Avicii");  
 stack.push("Jeepeta");  
 *assertEquals*(4, stack.size());  
}

Para esta prueba se lleno de 4 objetos el stack y luego se probo la función size para comprobar si devolvía la cantidad correcta de objetos en el stack.

1. Test de empty

@Test public void TestEmpty() {  
 *assertEquals*(true, stack.empty());  
 stack.push("Hola");  
 *assertEquals*(false, stack.empty());  
}

Para esta prueba se comprueba que este vacio el stack, para luego meterle un objeto y comprobar si el boolean de la función empty cambia de estado al meter un objeto

Las pruebas fueron superadas con éxito

