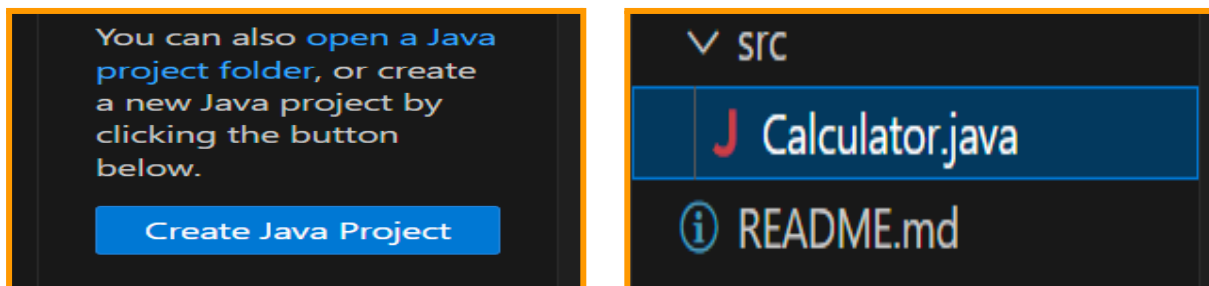


Tarea: Depuración y pruebas

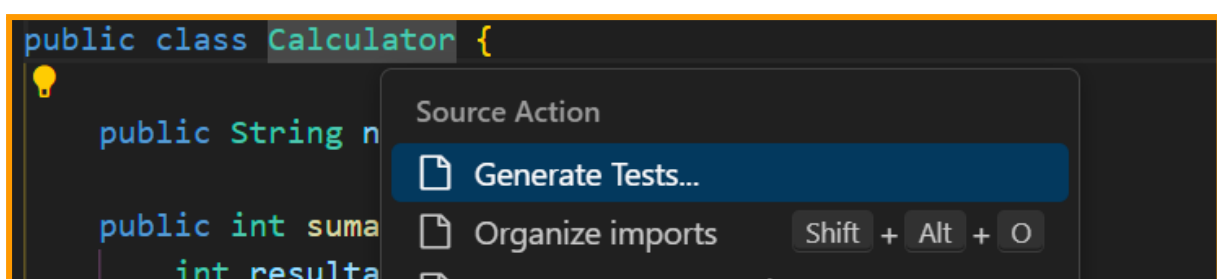
Ejercicio 1: Calculadora Simple

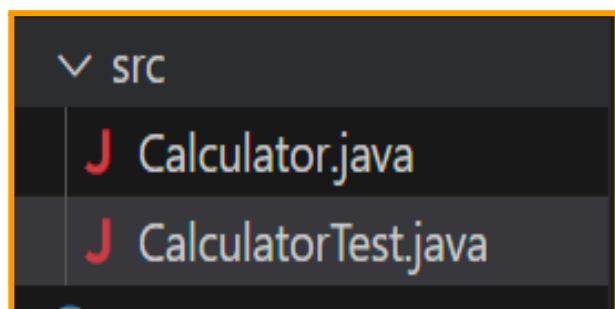
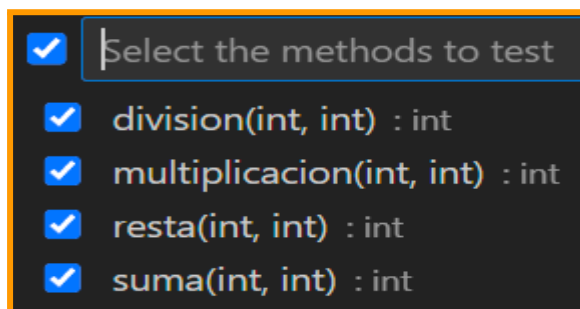
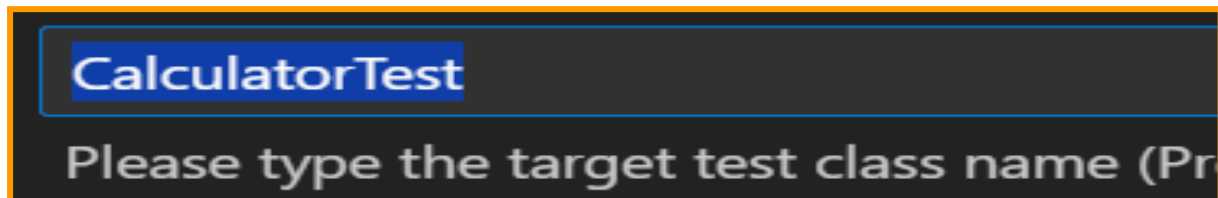
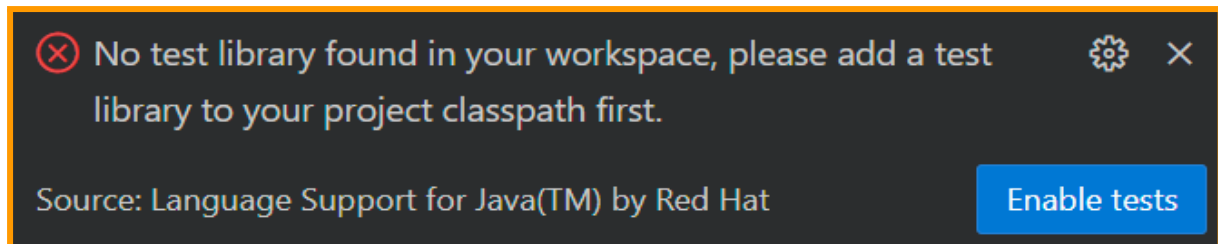
Lo primero que se debe hacer es crear un proyecto de java nuevo, en él se crea un archivo .java que será donde se escribirá el código del programa de la calculadora, con todos los métodos requeridos, así como nombre de atributos diferentes para los datos de los diferentes métodos para poder diferenciar los tests bien y de manera clara y concisa posteriormente.



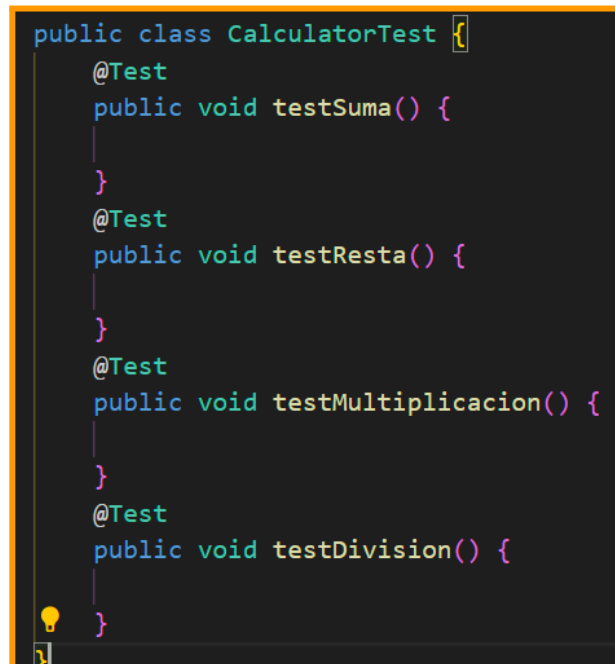
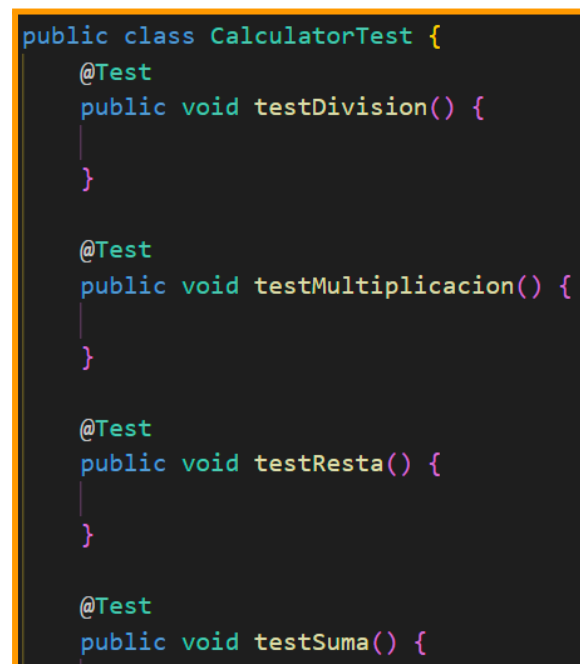
```
public class Calculator {  
    public String nombre = "";  
  
    public int suma(int sumando1, int sumando2) {  
        int resultado = sumando1 + sumando2;  
        return resultado;  
    }  
    public int resta(int minuendo, int sustraendo) {  
        int resultado = minuendo - sustraendo;  
        return resultado;  
    }  
    public int multiplicacion(int multiplicando1, int multiplicando2)  
    {  
        int resultado = multiplicando1 * multiplicando2;  
        return resultado;  
    }  
    public int division(int dividendo, int divisor) {  
        int resultado = dividendo / divisor;  
        return resultado;  
    }  
}
```

Una vez el programa de la calculadora con todos sus métodos esté completado se debe habilitar y generar los test, lo que dará lugar a un nuevo archivo .java, que se podrá nombrar, en el que se almacenarán los tests para los diferentes métodos previamente creados.





Dentro de este archivo deberá antes de nada ordenarse los diferentes tests ya automáticamente creados por Visual Studio según su orden real en el archivo de la calculadora, puesto que por defecto viene con otro orden y por tanto es menos intuitivo.



Una vez hecho eso, se procederá a elaborar el código para cada uno de los tests, el cual en este ejercicio concreto será muy similar para todos los métodos, ya que se trata simplemente de operaciones básicas. Para ello lo primero será dividir el proceso del test en 3 etapas: Arrange/given, la sección donde se establecen los datos que se usarán para realizar el test así como resultantes esperados en base a ellos o datos similares, además de inicializar la calculadora.

```
@Test
public void testSuma() {
    //Argange or given
    int sumando1 = 12;
    int sumando2 = -7;
    int resultadoEsperado = 5;
```

```
Calculator calculator = new Calculator();
```

Puesto que con este tipo de test solo se está comprobando una suma particular entre 2 valores establecidos, y no el funcionamiento de la calculadora en base a otros medios, sería preciso añadir en el nombre además una especificación del cálculo que se está haciendo, puesto que en caso de hacerse más tests de suma con otros valores diferentes, lo cual sería útil de hacer para comprobar que el método funciona correctamente y no ha sido coincidencia, no habría forma de diferenciar ambos tests.

```
@Test
public void testSuma_12_y_menos7() {
    //Argange or given
    int sumando1 = 12;
    int sumando2 = -7;
    int resultadoEsperado = 5;
```

(En base a esto deberían hacerse al menos 2 tests diferentes para cada método)

```
@Test
public void testSuma_2_y_3() {
    //Argange or given
    int sumando1 = 2;
    int sumando2 = 3;
    int resultadoEsperado = 5;
```

2º Etapa: Act/when, la parte del código donde se realiza el cálculo/comprobación del código original, debido a haber inicializado la calculadora previamente, ahora es posible utilizar el método suma del código original, usando como datos los números establecidos arriba, dando lugar a el resultado que el código original da como respuesta, obviando el resultado esperado.

```
//Act or when  
int resultadoReal = calculator.suma(sumando1, sumando2);
```

3º Etapa: Assert/then, donde se realiza la comprobación entre el resultado esperado, el que debería dar el programa si estuviese bien estructurado, y el resultado real que este ofrece.

```
//Assert or then  
Assert.assertEquals(resultadoEsperado, resultadoReal);
```

Para esta última etapa se utiliza un assert, es decir, métodos que trae la librería de JUnit que permiten comprobar si el resultado es correcto, debe ser importado.

```
Assert.assertEquals  
import org.junit.Assert;
```

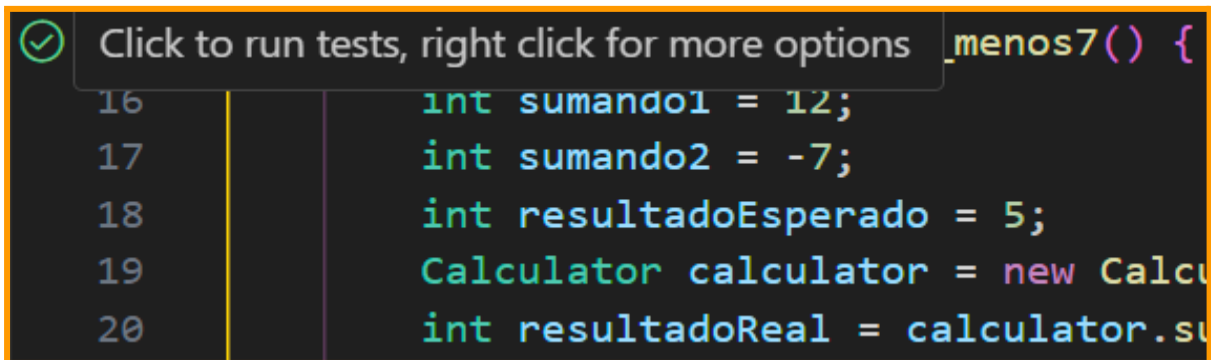
Una vez realizada la comprobación del código con el test, pudiendo hacerlo individualmente test a test, o globalmente todos a la vez, Visual Studio dará el resultado, en caso correcto, no ocurrirá nada y simplemente aparecerá un tick verde al lado del test indicando que es correcto, en caso de fallo, saldrá un aviso que mostrará que la comprobación a fallado, señalando cual es el resultado esperado en comparación con el que el programa a dado.

```
Click to run tests, right click for more options  
7  
8  
9  
10  
11  
12  
13  
int sumando1 = 2;  
int sumando2 = 3;  
int resultadoEsperado = 5;  
Calculator calculator = new Calculator();  
int resultadoReal = calculator.suma(sumando1, sumando2);  
Assert.assertEquals(resultadoEsperado, resultadoReal);  
}
```

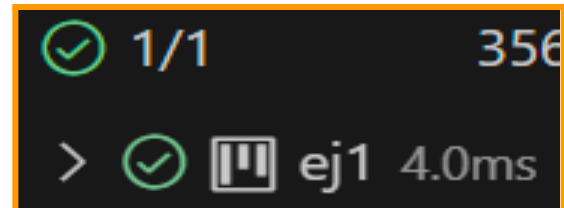
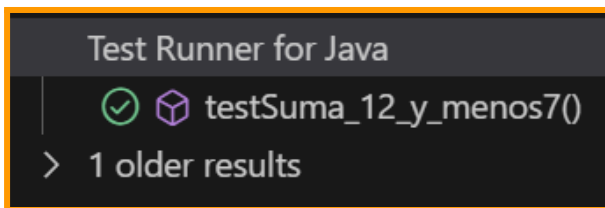
Test Global:

```
4  
5  
public class CalculatorTest {  
    @Test
```

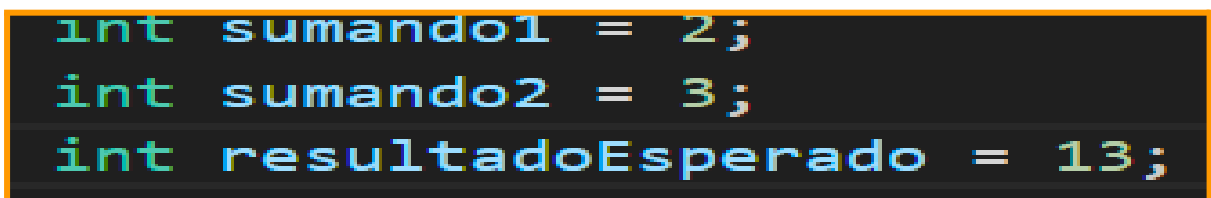
Test correcto:



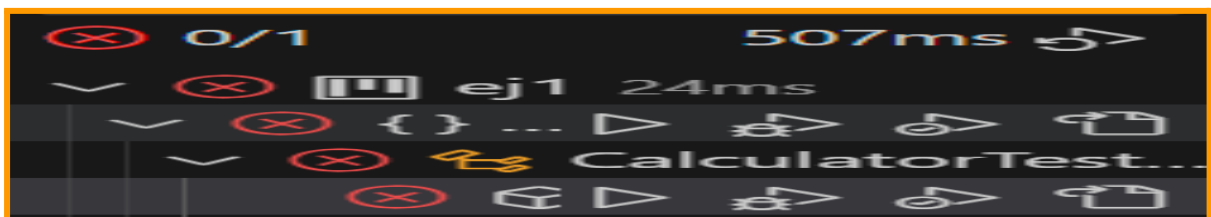
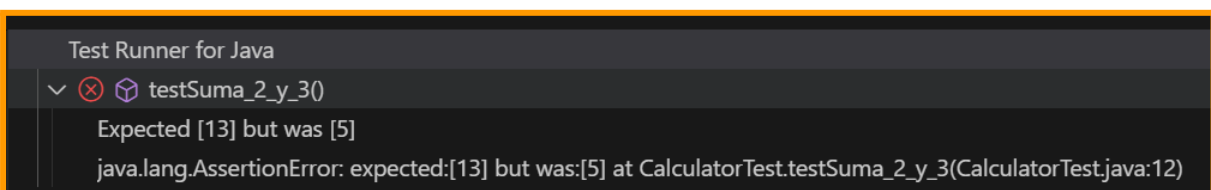
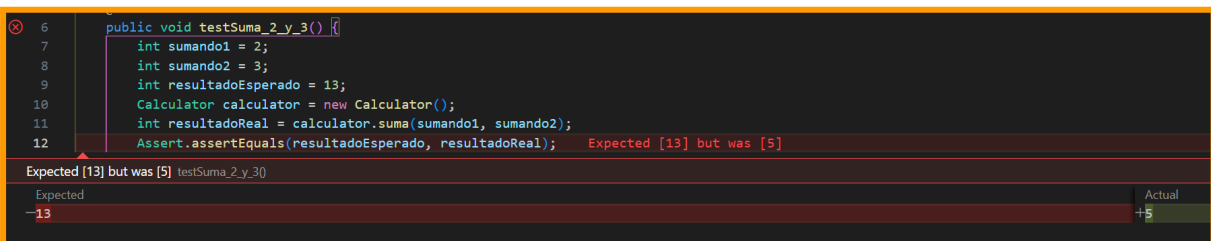
```
Click to run tests, right click for more options
16      int sumando1 = 12;
17      int sumando2 = -7;
18      int resultadoEsperado = 5;
19      Calculator calculator = new Calcula
20      int resultadoReal = calculator.su
```



Test fallido:



```
int sumando1 = 2;
int sumando2 = 3;
int resultadoEsperado = 13;
```



Si tras todo esto finalmente se comprueba mediante cada uno de los test que el código funciona en su completitud bien, habrá finalizado la fase de pruebas y el código estará listo.



Ejercicio 2: Testing y Debugging

Lo primero antes de nada es revisar los 3 archivos del código inicial, una vez hecho eso, se crean los 2 archivos con tests unitarios para ComparacionEnteros y para OperacionesMixtas, de la misma forma que en el ejercicio anterior, habilitando los tests, creando la carpeta...

```
J ComparacionesEnteros.java
J ComparacionesEnterosTest.java
J EjercicioTestingDebugging.java
J OperacionesMixtas.java
J OperacionesMixtasTest.java
```

Una vez generados ambos archivos, se diseñan los diferentes tests para cada uno de los métodos de ellos, de nuevo igual que en el ejercicio anterior, es decir, asignando valores de prueba así como el resultado real que el código debería ofrecer en base a ellos y comparándolo con el resultado real que el código genera en base a lo que hay escrito en él, para cada test, o por lo menos los más complejos, sería preciso realizar varios tests con diferentes valores cada uno. Al igual que en el anterior, en este caso solo se hará en base a unos únicos valores para agilizar el ejercicio.

```

import org.junit.Assert;
import org.junit.Test;

public class ComparacionesEnterosTest {

    @Test
    public void testSonIguales() {
        // Arrange or given
        int num1 = 2;
        int num2 = 2;
        boolean expectedResult = true;
        // Act or when
        boolean actualResult = ComparacionesEnteros.sonIguales(num1, num2);
        // Assert or then
        Assert.assertEquals(expectedResult, actualResult);
    }

    @Test
    public void testEsMayor() {
        // Arrange or given
        int num1 = 5;
        int num2 = 2;
        boolean expectedResult = true;
        // Act or when
        boolean actualResult = ComparacionesEnteros.esMayor(num1, num2);
        // Assert or then
        Assert.assertEquals(expectedResult, actualResult);
    }

    @Test
    public void testEsMenor() {
        // Arrange or given
        int num1 = 5;
        int num2 = 2;
        boolean expectedResult = false;
        // Act or when
        boolean actualResult = ComparacionesEnteros.esMenor(num1, num2);
        // Assert or then
        Assert.assertEquals(expectedResult, actualResult);
    }

    @Test
    public void testEsDivisible() {
        // Arrange or given
        int num1 = 10;
        int num2 = 0;
        boolean expectedResult = false;
        // Act or when
        boolean actualResult = ComparacionesEnteros.esDivisible(num1, num2);
        // Assert or then
        Assert.assertEquals(expectedResult, actualResult);
    }
}

```

```

@Test
public void testSonAmbosPares() {
    // Arrange or given
    int num1 = 5;
    int num2 = 2;
    boolean expectedResult = false;
    // Act or when
    boolean actualResult = ComparacionesEnteros.sonAmbosPares(num1, num2);
    // Assert or then
    Assert.assertEquals(expectedResult, actualResult);
}

@Test
public void testAlMenosUnoPositivo() {
    // Arrange or given
    int num1 = 5;
    int num2 = -2;
    boolean expectedResult = true;
    // Act or when
    boolean actualResult = ComparacionesEnteros.alMenosUnoPositivo(num1, num2);
    // Assert or then
    Assert.assertEquals(expectedResult, actualResult);
}

@Test
public void testSumaEsPar() {
    // Arrange or given
    int num1 = 5;
    int num2 = 2;
    boolean expectedResult = false;
    // Act or when
    boolean actualResult = ComparacionesEnteros.sumaEsPar(num1, num2);
    // Assert or then
    Assert.assertEquals(expectedResult, actualResult);
}

```

```

import org.junit.Assert;
import org.junit.Test;

public class OperacionesMixtasTest {
    @Test
    public void testConcatenarNumeroTexto() {
        // Arrange or given
        int numero = 5;
        String texto = "manzanas";
        String expectedResult = "5 manzanas";
        // Act or when
        String actualResult = OperacionesMixtas.concatenarNumeroTexto(numero, texto);
        // Assert or then
        Assert.assertEquals(expectedResult, actualResult);
    }

    @Test
    public void testBooleanComoTexto() {
        // Arrange or given
        boolean valor = true;
        String expectedResult = "verdadero";
        // Act or when
        String actualResult = OperacionesMixtas.booleanComoTexto(valor);
        // Assert or then
        Assert.assertEquals(expectedResult, actualResult);
    }

    @Test
    public void testConcatenarTextos() {
        // Arrange or given
        String texto1 = "Hola";
        String texto2 = "Mundo";
        String expectedResult = "Hola Mundo";
        // Act or when
        String actualResult = OperacionesMixtas.concatenarTextos(texto1, texto2);
        // Assert or then
        Assert.assertEquals(expectedResult, actualResult);
    }

    @Test
    public void testDescripcionNumero() {
        // Arrange or given
        int numero = 10;
        String expectedResult = "El número es positivo";
        // Act or when
        String actualResult = OperacionesMixtas.descripcionNumero(numero);
        // Assert or then
        Assert.assertEquals(expectedResult, actualResult);
    }
}

```

```

@Test
public void testCambiarTextoAMayusculas() {
    // Arrange or given
    String texto = "java";
    boolean aMayusculas = true;
    String expectedResult = "JAVA";
    // Act or when
    String actualResult = OperacionesMixtas.cambiarTextoAMayusculas(texto, aMayusculas);
    // Assert or then
    Assert.assertEquals(expectedResult, actualResult);
}

@Test
public void testDescripcionConFloat() {
    // Arrange or given
    float valor1 = 3.14159f;
    float valor2 = 2.71828f;
    String expectedResult = "Los valores son: 3.14€ y 2.72€";
    // Act or when
    String actualResult = OperacionesMixtas.descripcionConFloat(valor1, valor2);
    // Assert or then
    Assert.assertEquals(expectedResult, actualResult);
}

@Test
public void testFormatearNumero() {
    // Arrange or given
    float numero = 123.45678f;
    int decimales = 2;
    String expectedResult = "123.46";
    // Act or when
    String actualResult = OperacionesMixtas.formatearNumero(numero, decimales);
    // Assert or then
    Assert.assertEquals(expectedResult, actualResult);
}

```


Una vez que ya estén todos los diferentes test de cada uno de los métodos diseñados para cada uno de sus métodos, se ejecutan dichos test, pudiendo ver que tests son superados, significando esto que el resultado que se esperaba es el mismo que el que finalmente se genera, y también que tests no son superados.

```
4 public class ComparacionesEnterosTest {
5     @Test
6     public void testSonIguales() {
7         // Arrange or given
8         int num1 = 2;
9         int num2 = 2;
10        boolean expectedResult = true;
11        // Act or when
12        boolean actualResult = ComparacionesEnteros.sonIguales(num1, num2);
13        // Assert or then
14        Assert.assertEquals(expectedResult, actualResult); Expected [true] but was [false]

```

Expected	Actual
true	false

```
@Test
public void testEsMenor() {
    // Arrange or given
    int num1 = 5;
    int num2 = 2;
    boolean expectedResult = false;
    // Act or when
    boolean actualResult = ComparacionesEnteros.esMenor(num1, num2);
    // Assert or then
    Assert.assertEquals(expectedResult, actualResult); Expected [false] but was [true]
}

```

```
@Test
public void testEsDivisible() {
    // Arrange or given
    int num1 = 10;
    int num2 = 0;
    boolean expectedResult = false;
    // Act or when
    boolean actualResult = ComparacionesEnteros.esDivisible(num1, num2);
    // Assert or then
    Assert.assertEquals(expectedResult, actualResult); Expected [false] but was [true]
}

```

```
@Test
public void testSonAmbosPares() {
    // Arrange or given
    int num1 = 5;
    int num2 = 2;
    boolean expectedResult = false;
    // Act or when
    boolean actualResult = ComparacionesEnteros.sonAmbosPares(num1, num2);
    // Assert or then
    Assert.assertEquals(expectedResult, actualResult); Expected [false] but was [true]
}

```

```

@Test
public void testConcatenarNumeroTexto() {
    // Arrange or given
    int numero = 5;
    String texto = "manzanas";
    String expectedResult = "5 manzanas";
    // Act or when
    String actualResult = OperacionesMixtas.concatenarNumeroTexto(numero, texto);
    // Assert or then
    Assert.assertEquals(expectedResult, actualResult); Expected [5 manzanas] but was [5 5]
}
@Test

```

```

@Test
public void testConcatenarTextos() {
    // Arrange or given
    String texto1 = "Hola";
    String texto2 = "Mundo";
    String expectedResult = "Hola Mundo";
    // Act or when
    String actualResult = OperacionesMixtas.concatenarTextos(texto1, texto2);
    // Assert or then
    Assert.assertEquals(expectedResult, actualResult); Expected [Hola Mundo] but was [HolaMundo]
}

```

```

@Test
public void testDescripcionNumero() {
    // Arrange or given
    int numero = 10;
    String expectedResult = "El número es positivo";
    // Act or when
    String actualResult = OperacionesMixtas.descripcionNumero(numero);
    // Assert or then
    Assert.assertEquals(expectedResult, actualResult); Expected [El número es positivo] but was [El número es cero]
}

```

```

@Test
public void testDescripcionConFloat() {
    // Arrange or given
    float valor1 = 3.14159f;
    float valor2 = 2.71828f;
    String expectedResult = "Los valores son: 3.14€ y 2.72€";
    // Act or when
    String actualResult = OperacionesMixtas.descripcionConFloat(valor1, valor2);
    // Assert or then
    Assert.assertEquals(expectedResult, actualResult); Expected [Los valores son: 3.14€ y 2.72€] but was [Los valores son: 3.14159€ y 2.71828€]
}

```

Tras haber identificado los tests fallidos, para poder visualizar fácilmente por qué el código no da el resultado deseado, una forma de hacerlo sería hacer debugging en un archivo en donde los diferentes métodos sean llamados, en este caso EjercicioTestDebugging. Ahí añadiendo un breakpoint y haciendo debug, podrá verse, en base a unos datos preestablecidos, por qué el código no funciona de la forma deseada paso a paso, viendo a que datos cada método llama, los resultados que ofrece y el texto que escribe en consola.

```

1 public class EjercicioTestDebugging {
2     Run | Debug
3     public static void main(String[] args) { args = String[0]@9
4         int num1 = 4; num1 = 4
5         int num2 = 2; num2 = 2
6         // Ejecución de las funciones de la clase ComparacionesEnteros
7         System.out.println("Son iguales: " + ComparacionesEnteros.sonIguales(num1, num2)); num1 = 4, num2 = 2
8         System.out.println("Es mayor: " + ComparacionesEnteros.esMayor(num1, num2)); num1 = 4, num2 = 2
9         System.out.println("Es menor: " + ComparacionesEnteros.esMenor(num1, num2)); num1 = 4, num2 = 2
10        System.out.println("Es divisible: " + ComparacionesEnteros.esDivisible(num1, num2)); num1 = 4, num2 = 2
11        System.out.println("Son ambos pares: " + ComparacionesEnteros.sonAmbosPares(num1, num2)); num1 = 4, num2 = 2
12        System.out.println("Al menos uno positivo: " + ComparacionesEnteros.alMenosUnoPositivo(num1, num2)); num1 = 4, num2 = 2
13        System.out.println("Suma es par: " + ComparacionesEnteros.sumaEsPar(num1, num2)); num1 = 4, num2 = 2
14        // Ejecución de las funciones de la clase OperacionesMixtas
15        System.out.println(OperacionesMixtas.concatenarNumeroTexto(numero:5, texto:"manzanas"));
16        System.out.println(OperacionesMixtas.booleanComoTexto(valor:true));
17        System.out.println(OperacionesMixtas.concatenarTextos(texto1:"Hola", texto2:"Mundo"));
18        System.out.println(OperacionesMixtas.descripcionNumero(-10));
19        System.out.println(OperacionesMixtas.cambiarTextoAMayusculas(texto:"Java", aMayusculas:true));
20        System.out.println(OperacionesMixtas.descripcionConFloat(valor1:3.14f, valor2:2.71f));
21        System.out.println(OperacionesMixtas.formatearNumero(numero:123.456f, decimales:2));
22    }
23 }

```

```
Son iguales: true
Es mayor: true
Es menor: true
Es divisible: false
Son ambos pares: true
Al menos uno positivo: false
Suma es par: true
5 5
verdadero
HolaMundo
El número es positivo
JAVA
Los valores son: 3.14? y 2.71?
123.46
```

Además de hacer el debugging en base a unos solos datos, también podría intentarse usar otros valores diferentes y volver a hacer los tests, las veces que sean necesarias, comprobando así que el resultado de cada método no ha sido pura coincidencia y verdaderamente no funciona como debería. Una vez hecho el debugging, se habrá visto más claro cómo es que los diferentes métodos fallan y ya será viable dirigirse al código a corregirlo.

Diriéndonos primero al archivo ComporacionesEnteros, el primer método estaría mal puesto que en vez de ver si ambos números son iguales y devolver un true o false, comparaba si eran desiguales y en base a eso ofrecía el resultado.

```
// 1. Verifica si los dos números son iguales
public static boolean sonIguales(int a, int b) {
    boolean resultado = (a != b);
    return resultado;
}
```

Corregido:

```
// 1. Verifica si los dos números son iguales
public static boolean sonIguales(int a, int b) {
    boolean resultado = (a == b);
    return resultado;
}
```

El segundo método estaría correcto.

El tercer método estaría mal, ya que en lugar de verificar si el primer número es menor que el segundo, comprobaba si era mayor, al igual que en el anterior.

```
// 3. Verifica si el primer número es menor que el segundo
public static boolean esMenor(int a, int b) {
    boolean resultado = (a > b);
    return resultado;
}
```

Corregido:

```
// 3. Verifica si el primer número es menor que el segundo
public static boolean esMenor(int a, int b) {
    boolean resultado = (a < b);
    return resultado;
}
```

El cuarto método estaría mal, ya que devolvía true si el divisor era 0, ignorando que no se puede dividir por cero, y evaluaba incorrectamente el resto.

```
// 4. Verifica si el primer número es divisible por el segundo (sin resto)
public static boolean esDivisible(int a, int b) {
    // Primero verifica si el divisor es cero para evitar error
    boolean divisorEsCero = (b == 0);
    boolean resultado;
    if (divisorEsCero) {
        resultado = true;
    } else {
        int residuo = a % b;
        resultado = (residuo != 0);
    }
    return resultado;
}
```

Corregido:

```
// 4. Verifica si el primer número es divisible por el segundo (sin resto)
public static boolean esDivisible(int a, int b) {
    // Primero verifica si el divisor es cero para evitar error
    boolean divisorEsCero = (b == 0);
    boolean resultado;
    if (divisorEsCero) {
        resultado = false;
    } else {
        int residuo = a % b;
        resultado = (residuo == 0);
    }
    return resultado;
}
```

El quinto método estaría mal, ya que utilizaba || en lugar de &&, devolviendo true si cualquiera de los dos números era par, en lugar de verificar que ambos lo fueran.

```
// 5. Verifica si ambos números son pares
public static boolean sonAmbosPares(int a, int b) {
    boolean primerNumeroPar = (a % 2 == 0);
    boolean segundoNumeroPar = (b % 2 == 0);
    boolean resultado = primerNumeroPar || segundoNumeroPar;
    return resultado;
}
```

Corregido:

```
// 5. Verifica si ambos números son pares
public static boolean sonAmbosPares(int a, int b) {
    boolean primerNumeroPar = (a % 2 == 0);
    boolean segundoNumeroPar = (b % 2 == 0);
    boolean resultado = primerNumeroPar && segundoNumeroPar;
    return resultado;
}
```

El sexto método estaría mal, ya que utilizaba && en lugar de ||, devolviendo true solo si ambos números eran positivos en vez de si al menos uno lo era, además calcula mal si el segundo número es positivo.

```
// 6. Verifica si al menos uno de los dos números es positivo
public static boolean alMenosUnoPositivo(int a, int b) {
    boolean primerNumeroPositivo = (a > 0);
    boolean segundoNumeroPositivo = (b < 0);
    boolean resultado = primerNumeroPositivo && segundoNumeroPositivo;
    return resultado;
}
```

Corregido:

```
// 6. Verifica si al menos uno de los dos números es positivo
public static boolean alMenosUnoPositivo(int a, int b) {
    boolean primerNumeroPositivo = (a > 0);
    boolean segundoNumeroPositivo = (b < 0);
    boolean resultado = primerNumeroPositivo || segundoNumeroPositivo;
    return resultado;
}
```

El séptimo método estaría mal, ya que incrementaba los valores originales (++a y ++suma), alterando los datos de entrada y evaluando erróneamente si la suma era par.

```
// 7. Verifica si la suma de los dos números es par
public static boolean sumaEsPar(int a, int b) {
    int suma = ++a + b;
    boolean sumaPar = (++suma % 2 == 0);
    return sumaPar;
}
```

Corregido:

```
// 7. Verifica si la suma de los dos números es par
public static boolean sumaEsPar(int a, int b) {
    int suma = a + b;
    boolean sumaPar = (suma % 2 == 0);
    return sumaPar;
}
```

En el archivo OperacionesMixtas, el primer método estaría mal, ya que duplicaba el número en lugar de concatenarlo correctamente con el texto.

```
/*
 * 1. Concatena un número y una cadena de texto
 * dejando un espacio en medio
 * Ejemplo: concatenarNumeroTexto(5, "manzanas");
 * devuelve: "5 manzanas"
 */
public static String concatenarNumeroTexto(int numero, String texto) {
    String numeroComoTexto = Integer.toString(numero);
    String resultado = numeroComoTexto + " " + numero;
    return resultado;
}
```

Corregido:

```
public static String concatenarNumeroTexto(int numero, String texto) {
    String numeroComoTexto = Integer.toString(numero);
    String resultado = numeroComoTexto + " " + texto;
    return resultado;
}
```

El segundo método estaría mal, ya que siempre devolvía "verdadero", ignorando el valor booleano recibido.

```
/*
 * 2. Devuelve el String "verdadero" o "falso" dependiendo del valor booleano que recibe
 */
public static String booleanComoTexto(boolean valor) {
    String resultado;
    if (valor) {
        resultado = "verdadero";
    } else {
        resultado = "falso";
    }
    return resultado;
}
```

Corregido:

```
public static String booleanComoTexto(boolean valor) {
    String resultado;
    if (valor) {
        resultado = "verdadero";
    } else {
        resultado = "falso";
    }
    return resultado;
}
```

El tercer método estaría mal, ya que no añadía un espacio entre los textos concatenados.

```
/*
 * 3. Concatena dos cadenas de texto dejando un espacio en medio
 * Ejemplo: concatenarTextos("Hola", "Mundo");
 * devuelve: "Hola Mundo"
 */
public static String concatenarTextos(String texto1, String texto2) {
    String resultado = texto1 + "" + texto2;
    return resultado;
}
```

Corregido:

```
public static String concatenarTextos(String texto1, String texto2) {
    String resultado = texto1 + " " + texto2;
    return resultado;
}
```

El cuarto método estaría mal, ya que los mensajes estaban invertidos y la lógica para determinar si el número era cero era incorrecta.

```
/*
 * 4. Devuelve un mensaje sobre si el número dado es positivo, negativo o cero
 */
public static String descripcionNumero(int numero) {
    String resultado;
    if (numero < 0) {
        resultado = "El número es positivo";
    } else if (numero <= 0) {
        resultado = "El número es negativo";
    } else {
        resultado = "El número es cero";
    }
    return resultado;
}
```


Corregido:

```
public static String descripcionNumero(int numero) {
    String resultado;
    if (numero > 0) {
        resultado = "El número es positivo";
    } else if (numero < 0) {
        resultado = "El número es negativo";
    } else {
        resultado = "El número es cero";
    }
    return resultado;
}
```

El quinto método estaría correcto, ya que transforma adecuadamente el texto según el valor booleano recibido.

El sexto método estaría mal, ya que no limitaba los valores flotantes a dos decimales como lo indicaba la descripción.

```
/*
 * 6. Devuelve una descripción con dos valores flotantes concatenados
 * con dos decimales y el símbolo de Euro detrás
 * Ejemplo: (descripcionConFloat(3.14f, 2.71f);
 * devuelve: Los valores son: 3.14€ y 2.71€
 */
public static String descripcionConFloat(float valor1, float valor2) {
    String valor1ComoTexto = Float.toString(valor1) + "€";
    String valor2ComoTexto = Float.toString(valor2) + "€";
    String resultado = "Los valores son: " + valor1ComoTexto + " y " + valor2ComoTexto;
    return resultado;
}
```

Corregido:

```
public static String descripcionConFloat(float valor1, float valor2) {
    String valor1ComoTexto = String.format(format: "%.2f€", valor1);
    String valor2ComoTexto = String.format(format: "%.2f€", valor2);
    String resultado = "Los valores son: " + valor1ComoTexto + " y " + valor2ComoTexto;
    return resultado;
}
```

El séptimo método estaría correcto, ya que formatea adecuadamente el número flotante según los decimales indicados.

Una vez corregido el comportamiento de todos y cada uno de los métodos se podrían volver a hacer los tests unitarios y comprobar si ahora está bien el código, en caso de no estarlo se debería repetir el proceso, haciendo debugging y corrigiendo el código hasta que finalmente el resultado sea el esperado, entonces el programa estará plenamente funcional en base a lo previsto.

```
4 public class ComparacionesEnterosTest {
```

```
4 public class OperacionesMixtasTest {
```

```
1 public class EjercicioTestingDebugging {
2     Run | Debug
3     public static void main(String[] args) { args = String[0]@9
4         int num1 = 4; num1 = 4
5         int num2 = 2; num2 = 2
6         // Ejecución de las funciones de la clase ComparacionesEnteros
7         System.out.println("Son iguales: " + ComparacionesEnteros.sonIguales(num1, num2)); num1 = 4, num2 = 2
8         System.out.println("Es mayor: " + ComparacionesEnteros.esMayor(num1, num2)); num1 = 4, num2 = 2
9         System.out.println("Es menor: " + ComparacionesEnteros.esMenor(num1, num2)); num1 = 4, num2 = 2
10        System.out.println("Es divisible: " + ComparacionesEnteros.esDivisible(num1, num2)); num1 = 4, num2 = 2
11        System.out.println("Son ambos pares: " + ComparacionesEnteros.sonAmbosPares(num1, num2)); num1 = 4, num2 = 2
12        System.out.println("Al menos uno positivo: " + ComparacionesEnteros.alMenosUnoPositivo(num1, num2)); num1 = 4, num2 = 2
13        System.out.println("Suma es par: " + ComparacionesEnteros.sumaEsPar(num1, num2)); num1 = 4, num2 = 2
14        // Ejecución de las funciones de la clase OperacionesMixtas
15        System.out.println(OperacionesMixtas.concatenarNumeroTexto(numero:5, texto:"manzanas"));
16        System.out.println(OperacionesMixtas.booleanComoTexto(valor:true));
17        System.out.println(OperacionesMixtas.concatenarTextos(texto1:"Hola", texto2:"Mundo"));
18        System.out.println(OperacionesMixtas.descripcionNumero(-10));
19        System.out.println(OperacionesMixtas.cambiarTextoAMayusculas(texto:"Java", aMayusculas:true));
20        System.out.println(OperacionesMixtas.descripcionConFloat(valor1:3.14f, valor2:2.71f));
21        System.out.println(OperacionesMixtas.formatearNumero(numero:123.456f, decimales:2));
22    }
```

```
Son iguales: false
Es mayor: true
Es menor: false
Es divisible: true
Son ambos pares: true
Al menos uno positivo: true
Suma es par: true
5 manzanas
verdadero
Hola Mundo
El número es negativo
JAVA
Los valores son: 3.14? y 2.71?
123.46
```