

Examen

Ejercicio 3

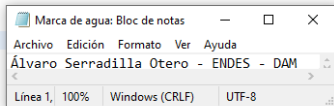
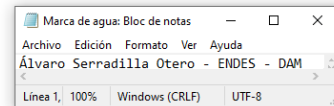
1) Realizar las pruebas correspondientes, utilizando JUnit para verificar estos métodos teniendo en cuenta que conocemos los siguientes datos:

Se crea un test para el ejercicio 3 par aprobar los dos métodos (Calcular_cantidad_solicitada y Calcular_interés_prestamo), tras esto introducimos los datos y realizamos las pruebas necesarias.

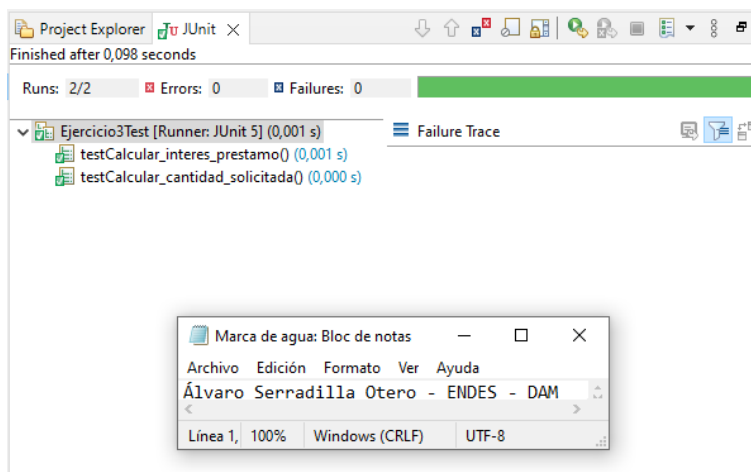
```
class Ejercicio3Test {

    @Test
    void testCalcular_cantidad_solicitada() {
        Ejercicio3 ejercicio = new Ejercicio3();
        double resultado1 = ejercicio.calcular_cantidad_solicitada(1000.00, 700.00);
        double valoresperado1 = 1000.0;
        assertEquals(resultado1, valoresperado1, 0);
        double resultado2 = ejercicio.calcular_cantidad_solicitada(4000.00, 700.00);
        double valoresperado2 = 3000.0;
        assertEquals(resultado2, valoresperado2, 0);
        double resultado3 = ejercicio.calcular_cantidad_solicitada(1000.00, -1000);
        double valoresperado3 = 0;
        assertEquals(resultado3, valoresperado3, 0);
        double resultado4 = ejercicio.calcular_cantidad_solicitada(18000.00, 1700.00);
        double valoresperado4 = 15000.0;
        assertEquals(resultado4, valoresperado4, 0);
    }

    @Test
    void testCalcular_interes_prestamo() {
        Ejercicio3 ejercicio = new Ejercicio3();
        double resultado1 = ejercicio.calcular_interes_prestamo(3500.00);
        double valoresperado1 = 6.0;
        assertEquals(resultado1, valoresperado1, 0);
        double resultado2 = ejercicio.calcular_interes_prestamo(7999.00);
        double valoresperado2 = 12.0;
        assertEquals(resultado2, valoresperado2, 0);
        double resultado3 = ejercicio.calcular_interes_prestamo(8001.00);
        double valoresperado3 = 13.5;
        assertEquals(resultado3, valoresperado3, 0);
        double resultado4 = ejercicio.calcular_interes_prestamo(-8001.00);
        double valoresperado4 = 0;
        assertEquals(resultado4, valoresperado4, 0);
    }
}
```



Tras esto lo ejecutamos para ver si tiene fallos o no :



2) Realizar las pruebas de cubrimiento del método “calcular_cantidad_solicitada”.

```
package ejercicio3;

public class Main {

    public static void main(String[] args) {
        Ejercicio3 ejercicio = new Ejercicio3();

        // Prueba 1
        double resultado1 = ejercicio.calcular_cantidad_solicitada(1000.00, 700.00);
        System.out.println("Prueba 1: Entrada (1000.00, 700.00), Salida esperada: 1000.0, Resultado: " + resultado1);

        // Prueba 2
        double resultado2 = ejercicio.calcular_cantidad_solicitada(1000.00, -1000);
        System.out.println("Prueba 2: Entrada (1000.00, -1000), Salida esperada: 0, Resultado: " + resultado2);

        // Prueba 3
        double resultado3 = ejercicio.calcular_cantidad_solicitada(18000.00, 1700.00);
        System.out.println("Prueba 3: Entrada (18000.00, 1700.00), Salida esperada: 15000.0, Resultado: " + resultado3);

        // Prueba 4
        double resultado4 = ejercicio.calcular_cantidad_solicitada(18000.00, 1000.00);
        System.out.println("Prueba 3: Entrada (18000.00, 1000.00), Salida esperada: 9000.0, Resultado: " + resultado4);
    }
}
```

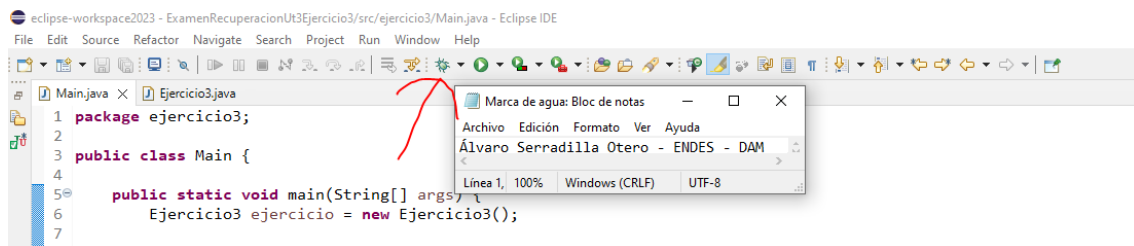
Con estas pruebas realizamos todas las pruebas de cubrimiento teniendo todas las posibilidades del método cubiertas y funcionando.

3) Depura la aplicación para los casos siguientes:

Ponemos puntos de ruptura en el código main para poder depurar el método calcular_cantidad_solicitada de los siguientes casos

```
50 public static void main(String[] args) {
51     Ejercicio3 ejercicio = new Ejercicio3();
52
53     // Prueba 1
54     double resultado1 = ejercicio.calcular_cantidad_solicitada(1000.00, 700.00);
55     System.out.println("Prueba 1: Entrada (1000.00, 700.00), Salida esperada: 1000.0, Resultado: " + resultado1);
56
57     // Prueba 2
58     double resultado2 = ejercicio.calcular_cantidad_solicitada(1000.00, -1000);
59     System.out.println("Prueba 2: Entrada (1000.00, -1000), Salida esperada: 0, Resultado: " + resultado2);
60
61     // Prueba 3
62     double resultado3 = ejercicio.calcular_cantidad_solicitada(18000.00, 1700.00);
63     System.out.println("Prueba 3: Entrada (18000.00, 1700.00), Salida esperada: 15000.0, Resultado: " + resultado3);
64 }
```

Tras esto iniciamos el depurador:



Ejercicio 4

1) Diseñar un caso de prueba no válido que tenga dos clases de equivalencia no válidos y las demás que sí lo sean.

Origen	Destino	Fecha de salida	Clase de vuelo	Número de pasajeros
Madrid	París	04/2023	Directiva	2

Nos mostraría dos tipos de errores siendo uno específico ya que la clase de vuelo no es apta para la búsqueda del programa y un error de la fecha ya que al no especificar el día de salida no nos podrá mostrar resultados de vuelo.

2) Diseñar 3 casos de prueba no válidos.

	Origen	Destino	Fecha de salida	Clase de vuelo	Número de pasajeros
Caso 1	Talavera de la reina	Madrid	23/05/2023	económica	3
Caso 2	Barcelona	Londres	05/04/2023	Ultima clase	2
Caso 3	Madrid	Tenerife	02/05/2023	ejecutiva	0

El error del caso 1 se encuentra en el trayecto que realiza el vuelo ya que no es posible por el hecho de que no salen vuelos desde dicho origen.

El error del caso 2 se encuentra en la clase de vuelo ya que no dicho tipo de clase no es válido.

El error del caso 3 se encuentra en el número de pasajeros ya que no es posible comprar un vuelo para ningún pasajero.

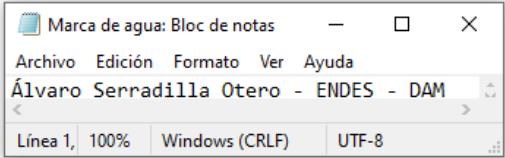
3) Diseñar 3 casos de prueba válidos para cada clase de equivalencia, incluyendo diferentes combinaciones de datos de entrada.

	Origen	Destino	Fecha de salida	Clase de vuelo	Número de pasajeros
Caso 1	Madrid	París	23/05/2023	económica	3
Caso 2	Barcelona	Londres	05/04/2023	Primera clase	2
Caso 3	Madrid	Tenerife	02/05/2023	ejecutiva	1

Ejercicio 5

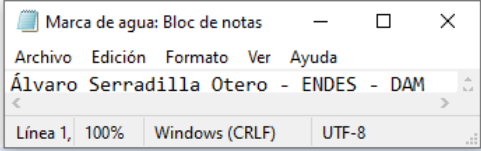
1) Encuentra los errores de compilación que hay en el código fuente y arrégloslos para que el programa funcione correctamente. Muestra, con capturas de pantalla, qué errores son y cómo los arreglaste.

```
public static String identificarNumero(int numero) {
    if (numero < 0) {{
        return "positivo";
    } else if (numero > 0) {
        return "negativo";
    } else {
        return "cero";
        return;
    }
}
```



Se puede ver como en esta parte nos encontramos con dos errores de compilación, en este caso para solucionarlo debemos quitar un corchete en el primer if y en el ultimo else quitar el return sin nada.

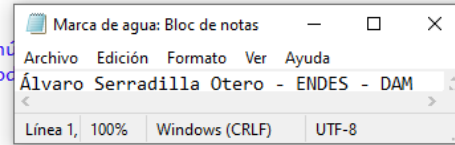
```
public static String identificarNumero(int numero) {
    if (numero < 0) {
        return "positivo";
    } else if (numero > 0) {
        return "negativo";
    } else {
        return "cero";
    }
}
```



```
do {{
    System.out.println("Por favor, introduce un número entre -1000 y 1000:");
    while (!scanner.hasNextInt()) {
        String input = scanner.next();
        System.out.printf("%s\n" no es un número válido.\n", input);
        System.out.println("Por favor, introduce un número válido.");
    }
    numero = scanner.nextInt();
} while (numero < -1000 || numero > 1000);

String resultado = identificarNumero(numero);
System.out.println("El número " + numero + " es: " + resultado);

scanner.close();
}
```

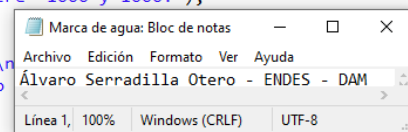


Aquí podemos ver como otro error de compilación donde en el do se puede ver que tiene un corchete mas, el cual para solucionarlo solo debemos eliminar ese corchete de sobra.

```
do {
    System.out.println("Por favor, introduce un número entre -1000 y 1000:");
    while (!scanner.hasNextInt()) {
        String input = scanner.next();
        System.out.printf("%s\n" no es un número válido.\n", input);
        System.out.println("Por favor, introduce un número válido.");
    }
    numero = scanner.nextInt();
} while (numero < -1000 || numero > 1000);

String resultado = identificarNumero(numero);
System.out.println("El número " + numero + " es: " + resultado);

scanner.close();
}
```



2) Explica la diferencia entre error de compilación y error lógico.

Los errores de compilación son aquellos errores mas básico que se pueden tener a la hora de crear el código, dichos errores son en base errores de la sintaxis del código, los cuales en muchas ocasiones te los marca el propio entorno de desarrollo que utilices para programar, en cambio los errores lógicos son mas complejos, ya en muchas ocasiones los propios entornos de desarrollo no los marcara como sin fueran fallos, estos errores podrían ser como un método que se encarga de realizar una suma realizara una resta.

3) Depura el código fuente y encuentra los errores lógicos que hay en el código.

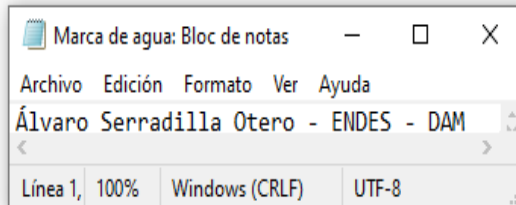
Para detectar donde se encontraban los problemas del código he ejecutado un par de veces el código completo.

Teniendo así dos resultados inesperados:

Por favor, introduce un número entre -1000 y 1000:

-23

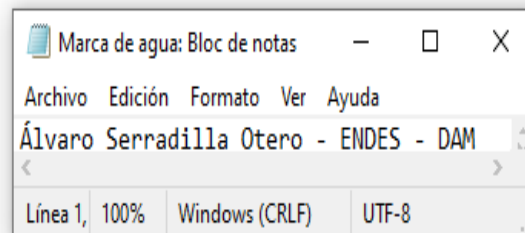
El número -23 es: positivo



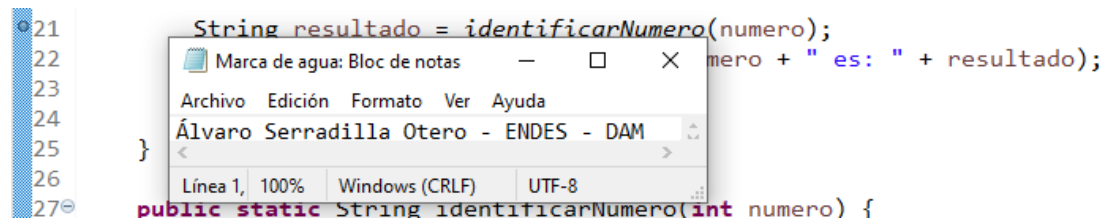
Por favor, introduce un número entre -1000 y 1000:

13

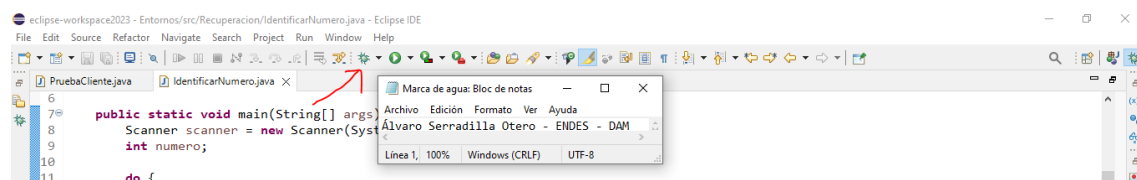
El número 13 es: negativo



Tras esto decido depurar el código para poder ver así donde está el problema poniendo un punto de ruptura en el código justo cuando se realiza el método de este :



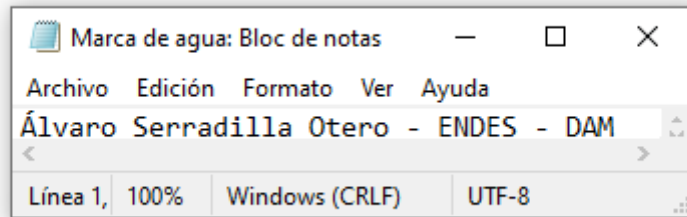
Con esto ya echo solo necesito empezar a depurar el código y buscar cual es el problema en este, para esto le daremos a este botón.



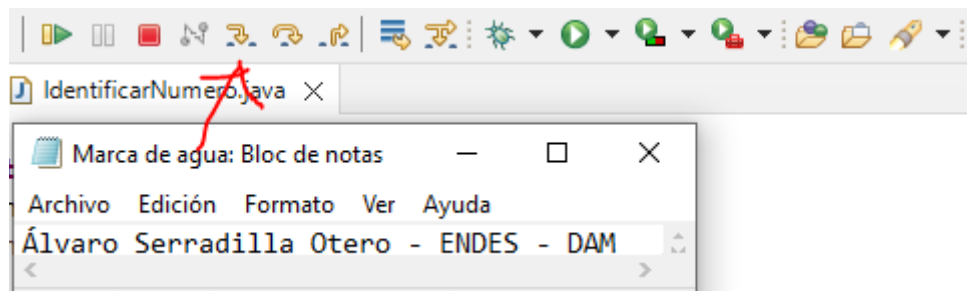
Y seguido de esto introduciremos un numero par nuestra prueba.

Por favor, introduce un número entre -1000 y 1000:

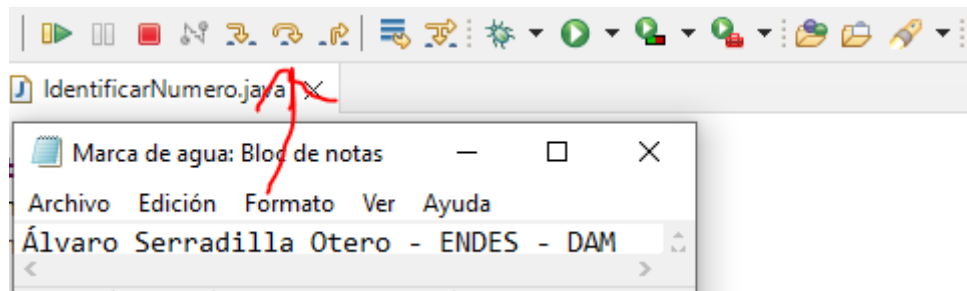
12



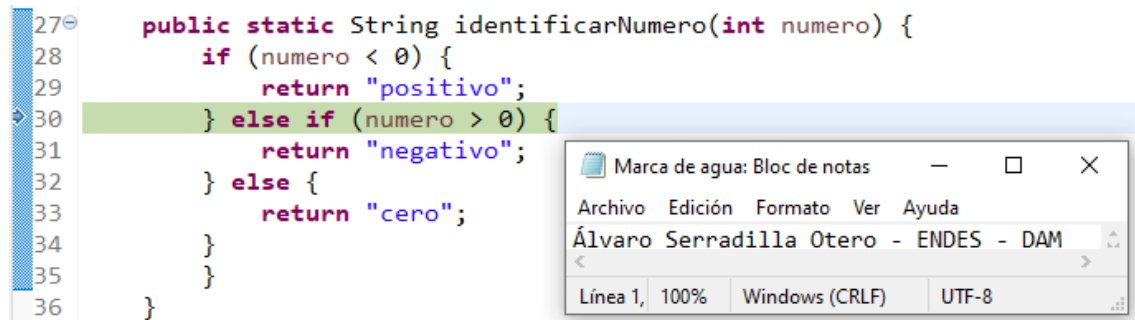
Y después con el depurador nos meteremos dentro del método para ver que es lo que ocurre, utilizando este botón :



Tras esto utilizaremos este botón para ver cada linea de código :

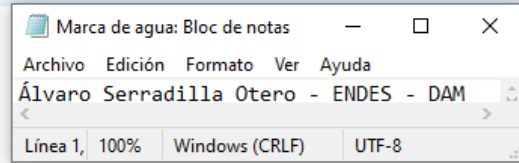


Con esto podemos ver que tras ver el if de si es positivo este se lo salta para ir directamente al else :



Tras esto nos damos cuenta que el problema se encuentra en las condiciones de los if las cuales están al revés, lo cual cambiamos para corregir dichos errores lógicos :

```
public static String identificarNumero(int numero) {
    if (numero > 0) {
        return "positivo";
    } else if (numero < 0) {
        return "negativo";
    } else {
        return "cero";
    }
}
```

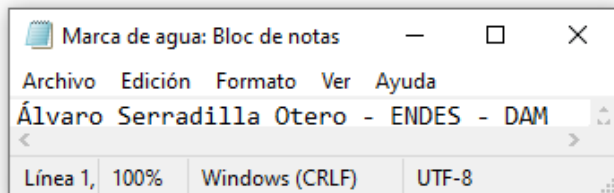


Y con esto ya solucionado tendremos los mismos resultados que los que dicta el enunciado :

Por favor, introduce un número entre -1000 y 1000:

500

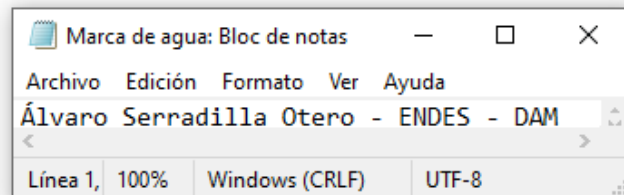
El número 500 es: positivo



Por favor, introduce un número entre -1000 y 1000:

-10

El número -10 es: negativo



Por favor, introduce un número entre -1000 y 1000:

0

El número 0 es: cero

