1. Debug Invertir Array

```java

package arrays;

public class invertir_array {
	public invertir_array() {
		// TODO Auto-generated constructor stub
	}
	public static void main(String[] args) {
	// TODO Auto-generated method stub
	//declaramos array a invertir y la variable que nos va a servir para
	//intercambiar los valores del array
	int array[] = {1,2,3,4,5,6,7,8,9};
	int aux=0;
	// imprimimos array para comprobar
	int longitud=array.length;
	for (int i = 0; i<=longitud; i++)
	 System.out.print(array[i] + " ");
	// invertimos array recorriéndolo hasta la mitad e intercambiando valores
	for(int i=0;i<=longitud/2; i++){
	//vamos guardando la posición que vamos a intercambiar desde el final
	 aux = array[longitud-i];
	 //como ya tenemos guardada la posición longiutd-i-1, ya podemos guardar en
	//esa posición un valor
	 array[longitud-i-1] = array[i];
	 //guardamos en las posiciones iniciales el valor de aux que ya estaba
	//guardado
	 array[i] = aux;
	 }
	// comprobamos que se ha invertido el array
	System.out.println();
	for (int i = 0; i < longitud; i++)
	 System.out.print(array[i] + " ");
	}
	}


```

Ilustración 1 Programa sin depurar

Name	Value
print() returned	(No explicit return value)
args	String[0] (id=20)
array	(id=21)
aux	0
i	8

Ilustración 2 Valores en depuración

Name	Value
no method return value	
args	String[0] (id=20)
array	(id=21)
aux	0
i	9

Ilustración 3 Valores en depuración 2

Cambios realizados:

1. Se ha eliminado la variable longitud porque no era necesaria, el trabajo se podía realizar con la propiedad .length.

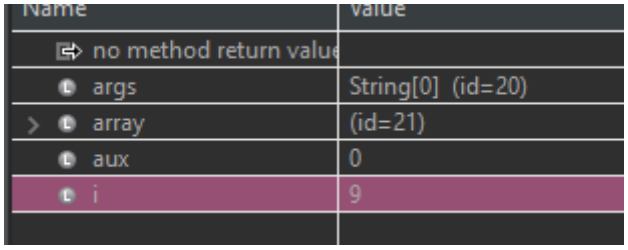


Ilustración 4 Valor i fuera límite

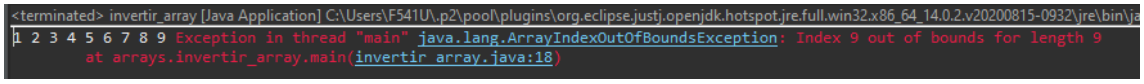
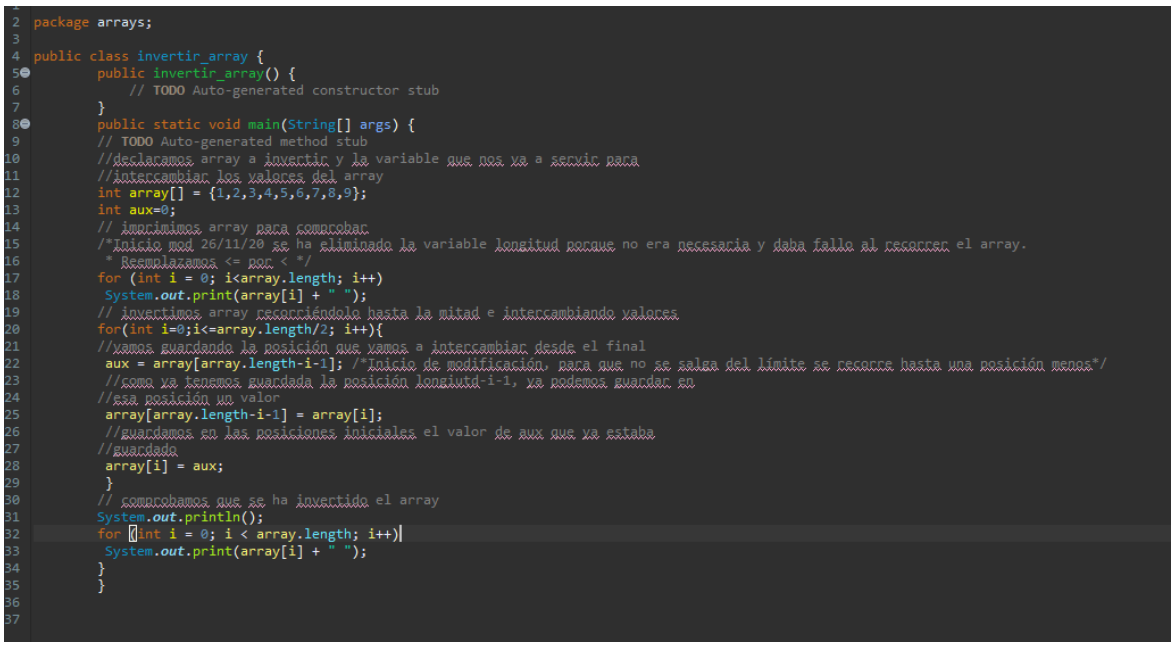


Ilustración 5 Mensaje de error

2. Al recorrer el array en la línea 17 saltaba un error de límites porque la condición de i era <=, por lo tanto, se ha cambiado por <.
3. En la línea 22 había otro error de límites que se ha solucionado añadiendo -1.

A continuación, se observan las capturas de los cambios realizados:



```
package arrays;

public class invertir_array {
        public invertir_array() {
            // TODO Auto-generated constructor stub
        }
        public static void main(String[] args) {
        // TODO Auto-generated method stub
        //declaramos array a invertir y la variable que nos va a servir para
        //intercambiar los valores del array
        int array[] = {1,2,3,4,5,6,7,8,9};
        int aux=0;
        // imprimimos array para comprobar
        /*Inicio mod 26/11/20 se ha eliminado la variable longitud porque no era necesaria y daba fallo al recorrer el array.
         * Reemplazamos <= por < */
        for (int i = 0; i<array.length; i++)
         System.out.print(array[i] + " ");
        // invertimos array recorriéndolo hasta la mitad e intercambiando valores
        for(int i=0;i<=array.length/2; i++){
        //vamos guardando la posición que vamos a intercambiar desde el final
         aux = array[array.length-i-1]; /*Inicio de modificación, para que no se salga del límite se recorre hasta una posición menos*/
         //como ya tenemos guardada la posición longitud-i-1, ya podemos guardar en
        //esa posición un valor
         array[array.length-i-1] = array[i];
         //guardamos en las posiciones iniciales el valor de aux que ya estaba
        //guardado
         array[i] = aux;
         }
        // comprobamos que se ha invertido el array
        System.out.println();
        for (int i = 0; i < array.length; i++)
         System.out.print(array[i] + " ");
        }
        }
```

Ilustración 6 Programa sin errores

Vemos cómo ahora el programa funciona correctamente, el array se invierte.

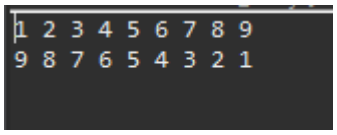


Ilustración 7 Array invertido

Junit test

Lo primero que tenemos que hacer para hacer una prueba de Junit es crear las clases que queremos probar, en este caso, primos y menorMayor.

```
public static int [] primos (int num) {

    //boolean esPrimo; //Creamos boolean para comprobar si se trata de un número primo
    List<Integer> listaPrimos = new ArrayList<Integer>();
    int arrayPrimos [] = new int [3];

    // Recorro el array, comprobando si cada elemento es un número primo
            for (int i = 1; i <= num; i++) {
                //if (i == 3) {
                    //listaPrimos.add(i);
                //}
                boolean esPrimo = true; // Utilizo una bandera, asumiendo que el número es primo
                // En el bucle siguiente intento demostrar que hay algún divisor del número.
                for (int j = 2; j < i - 1; j++) {
                    if (i % j == 0) { // j es divisor de numeros[i]
                        esPrimo = false; // Bajo la bandera, no hay primo
                    }
                }
                // Si salimos del bucle con la bandera arriba, el número es primo
                if (esPrimo == true) {
                    listaPrimos.add(i);
                }
            }
            //}
    for (int i  = 0; i < arrayPrimos.length; i++) {
        arrayPrimos[i] = listaPrimos.get(listaPrimos.size()-(1+i));

    }

    return arrayPrimos;
```

Ilustración 8 Método primos

```
public static int [] menorMayor (int array[]) {


    boolean hayIntercambios;
    do {
        hayIntercambios = false;
        // Empieza el algoritmo
        for (int i = 0; i < array.length - 1; i++) {
            if (array[i+1] < array[i]) {
                // Entonces hago un intercambio
                int aux = array[i+1];
                array[i+1] = array[i];
                array[i] = aux;
                hayIntercambios = true;
            }
        }
    } while (hayIntercambios);

    return array;
}
```

Ilustración 9 Método ordenación menor a mayor

Para comprobar que nuestros métodos funcionan usaremos la herramienta de Junit:

```
@Test
public void testPrimos () {
    int resultado []= menorMayor_primo.primos(12);
    int esperado [] = new int [] {11, 7, 5};

    assertArrayEquals(esperado, resultado);
}
```

Ilustración 10 Comprobación método primos

Comprobamos que nuestro método funciona si resultado y esperado coinciden. Lo mismo haremos con el segundo método, “menorMayor”.

```
@Test
public void testMenorMayor () {
    int array [] = new int [] {8, 6, 12};
    int resultado [] = menorMayor_primo.menorMayor(array);
    int esperado [] = new int [] {6, 8, 12};
}
```

Ilustración 11 Comprobación método menorMayor

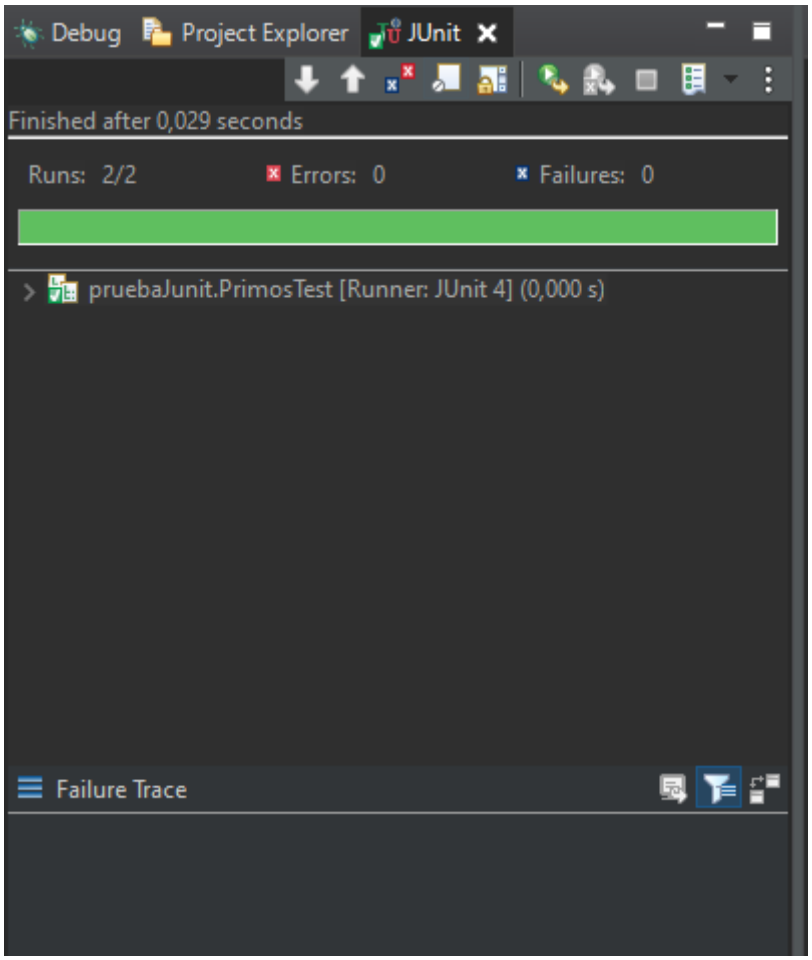


Ilustración 12 Comprobación fallos Junit

Si el método no funcionara correctamente, saltarían los errores para poder subsanarlos. Junit los muestra de la siguiente manera:

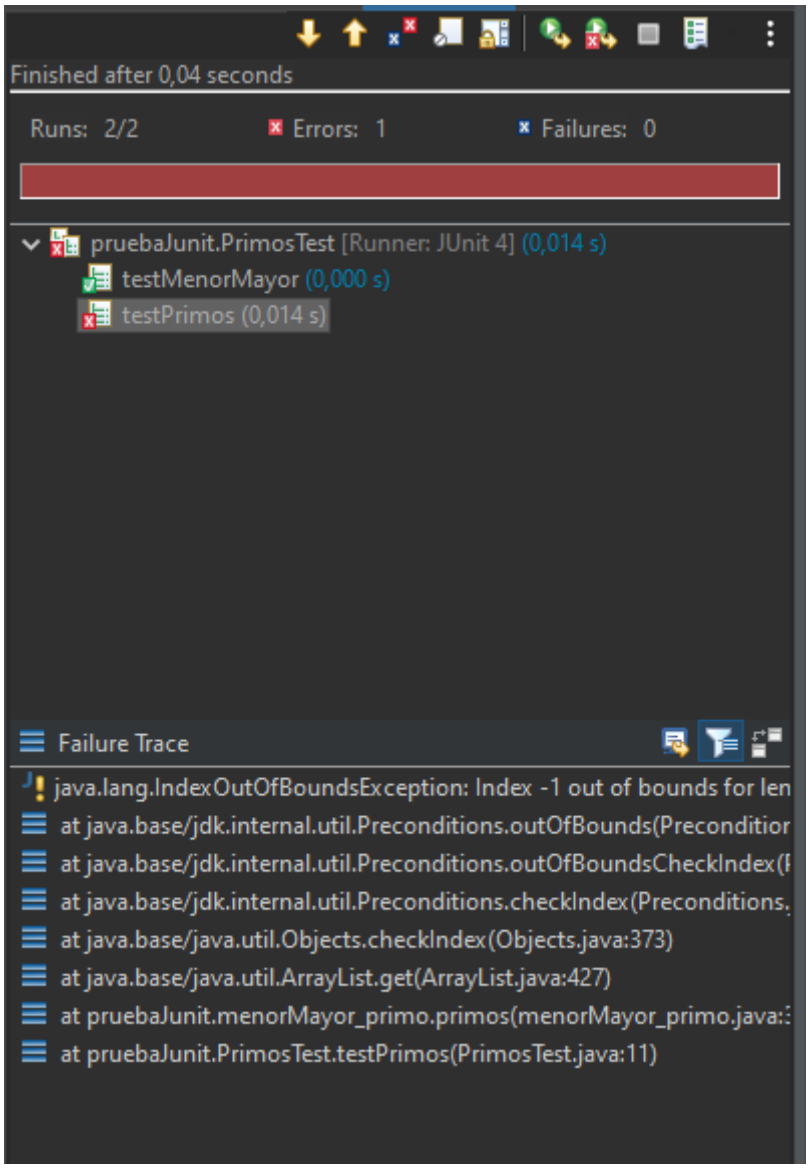


Ilustración 13 Ejemplo de error Junit