

TAREA DEL TEMA 3

ETS

07/05/2023

Autor: Derimán Tejera Fumero

ÍNDICE

Actividad 1.....	1
Actividad 2.....	8

Actividad 1: Utilizando NetBeans, deberás depurar el código que se presenta bajo el enunciado de la actividad. Encuentra cuáles son las instrucciones que están ocasionando problemas (tanto a nivel de compilación, como a nivel de ejecución), y corrégelas. Recuerda personalizar el entorno y mostrar claramente dónde están los fallos, qué pasos/herramientas has seguido para detectarlos y cómo has solucionado el problema. Muestra que funciona.

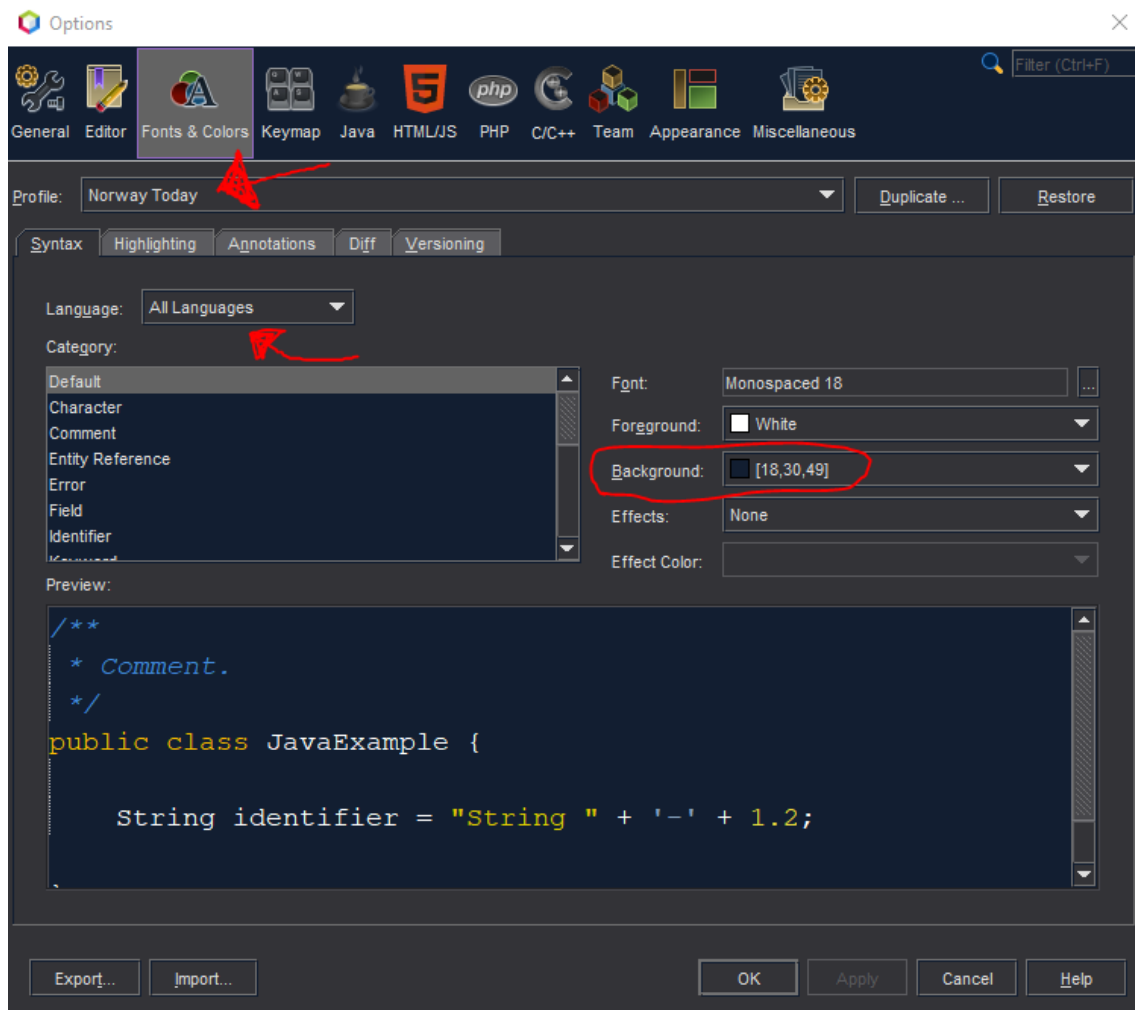
Código que del siguiente String “La lluvia en Sevilla es una maravilla” cuenta cuántas vocales hay en total (recorre el String con charAt). Este código incluirá errores de compilación que deben ser reparados antes de que podamos proceder con el depurado.

Actividad 1

```
public class Prueba {  
    public static void main(String[] args) {  
        String cadena="La lluvia en Sevilla es una maravilla";  
        int contador=0;  
        for (int i=1;i<cadena.length();i++){  
            //Comprobamos si el caracter es una vocal  
            if(cadena.charAt(i)=='a' || cadena.charAt(i)=='e' || cadena.charAt(i)=='i' || cadena.charAt(i)=='o' || cadena.charAt(i)=='u'){  
                System.out.println("Encontramos una vocal");  
            }  
            contador++;  
        }  
        System.out.println("Hay "+ contador +" vocales");  
    }  
}
```

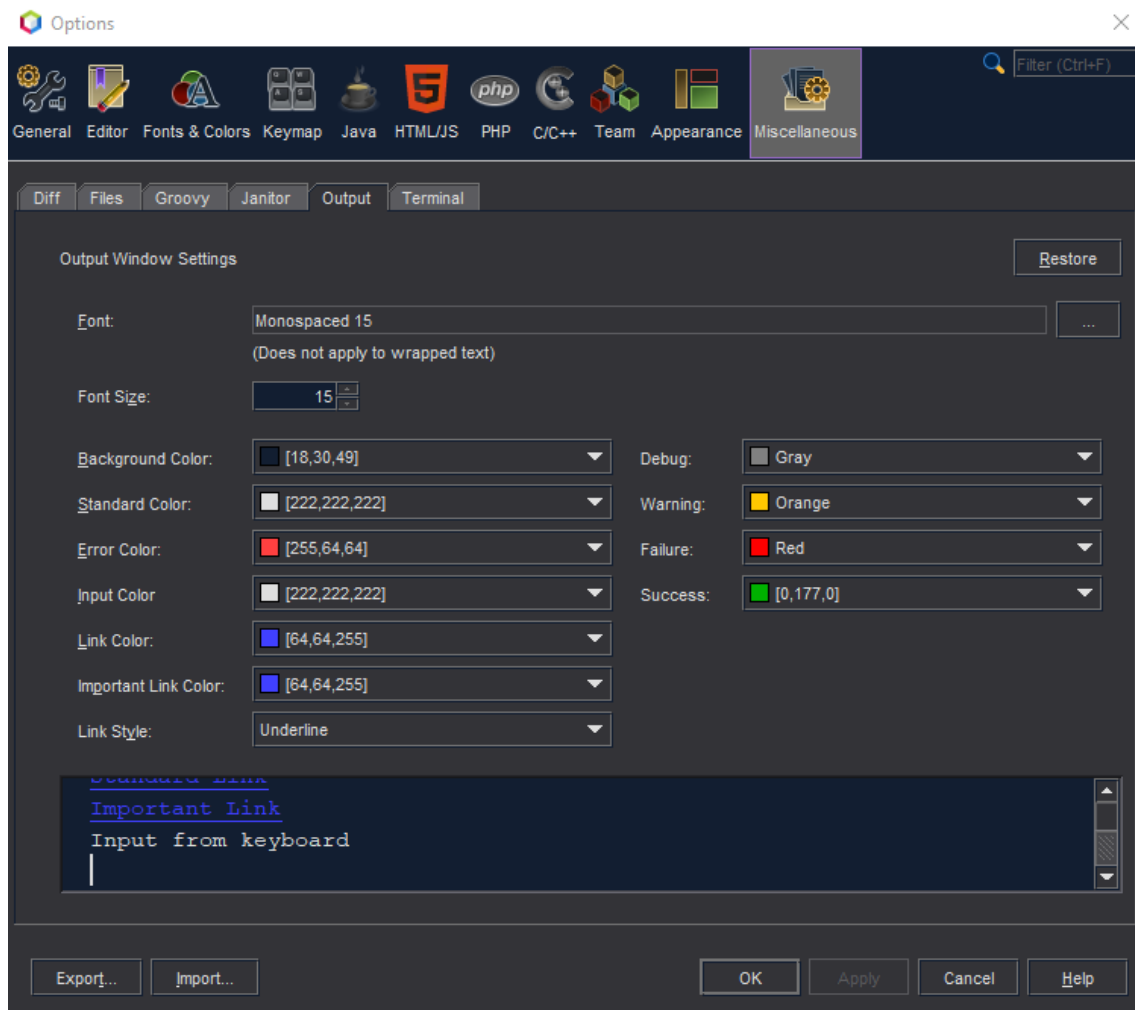
Las modificaciones hechas en NetBeans:

El estilo general del IDE lo he configurado para que sea Norway Today para todos los lenguajes que soporta el IDE, ya que es mucho mas cómodo a la vista a la hora de pasar muchas horas trabajando en código, al menos mucho mas que el blanco que viene por defecto:

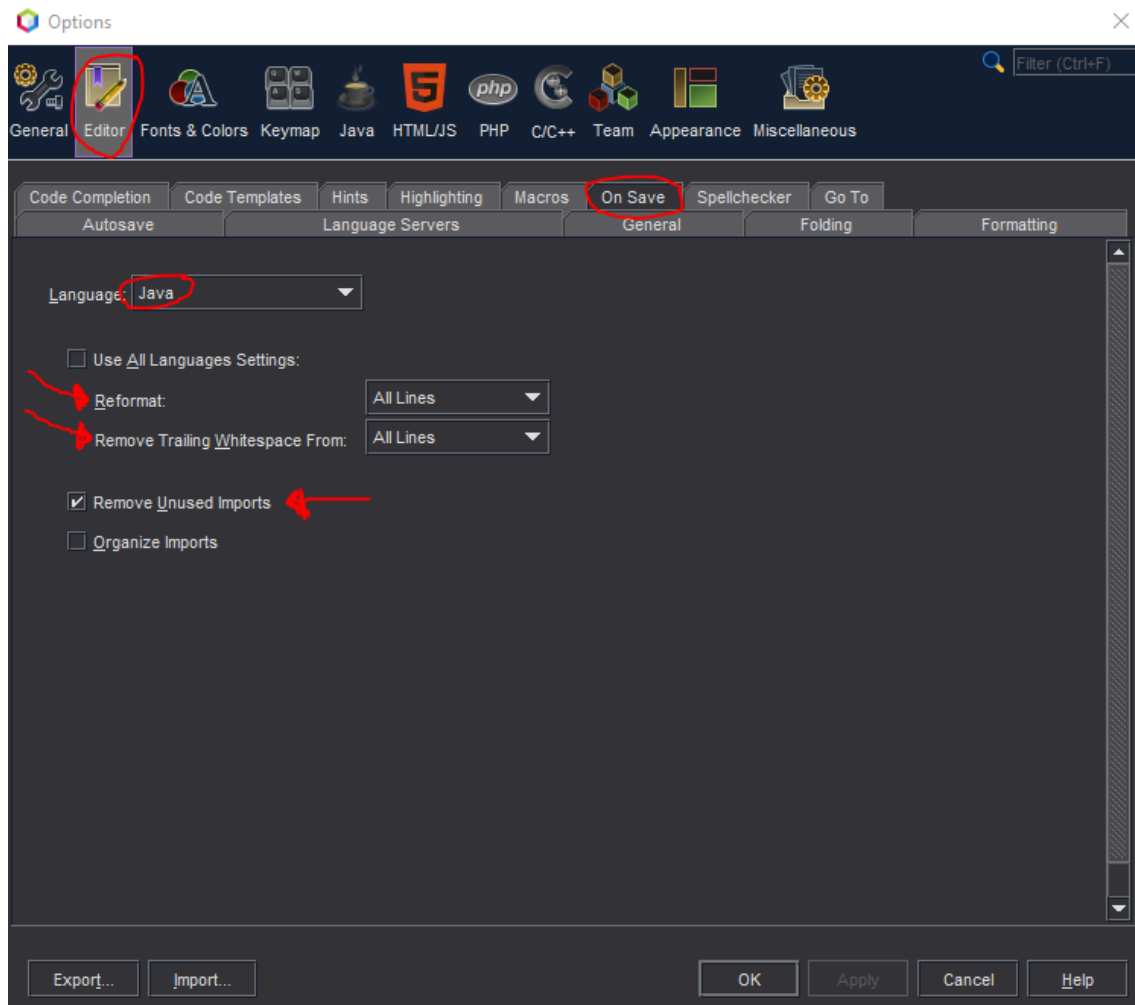


El resto de parámetros no los he modificado, los colores que ya incorpora Norway Today son bastante buenos y llamativos.

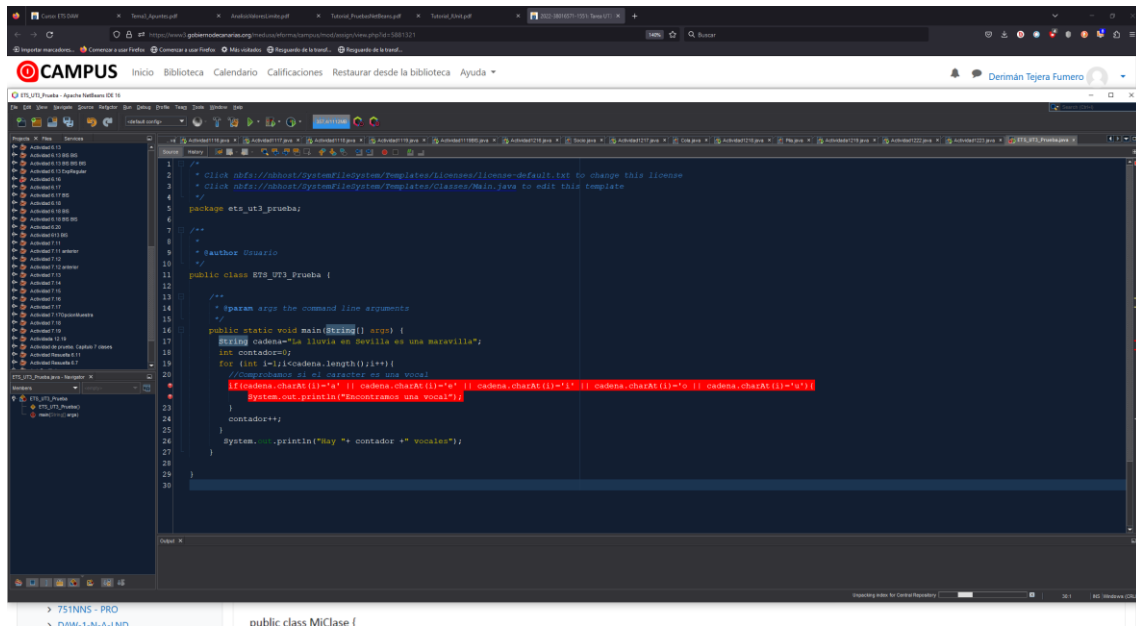
Para modificar el aspecto al mostrar errores, entramos en Tools > Options > Miscellaneous > Output



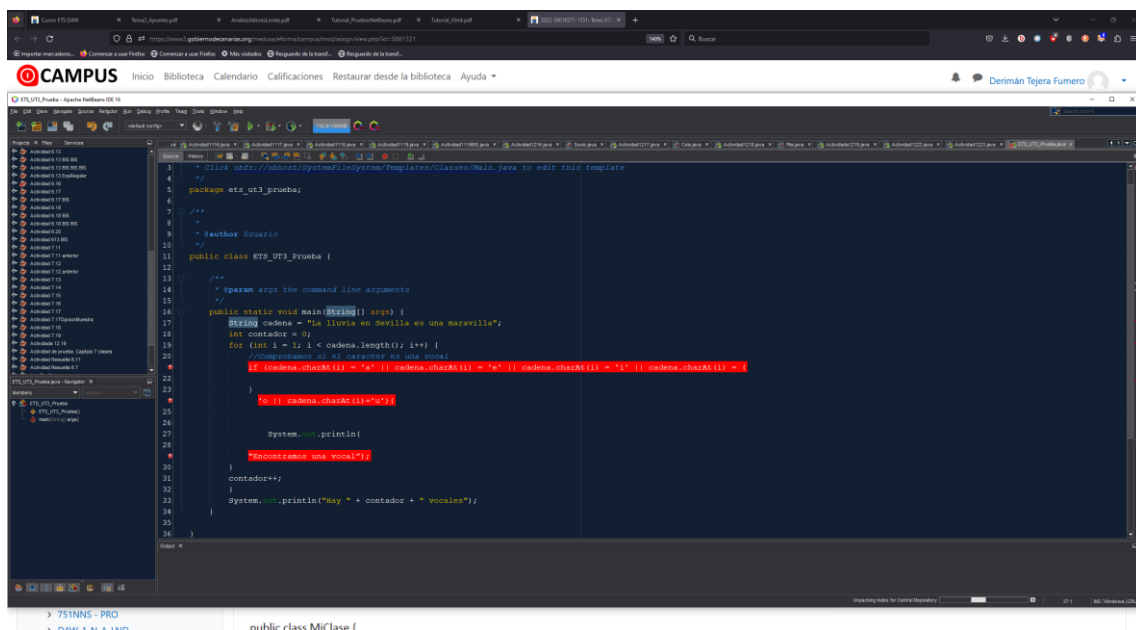
Para modificar el indentación (Reformat) en Java para que la corrija cada vez que guarde el archivo, y además elimine los imports que no sean necesarios, así como los espacios innecesarios(Remove Trailing Whitespaces From:



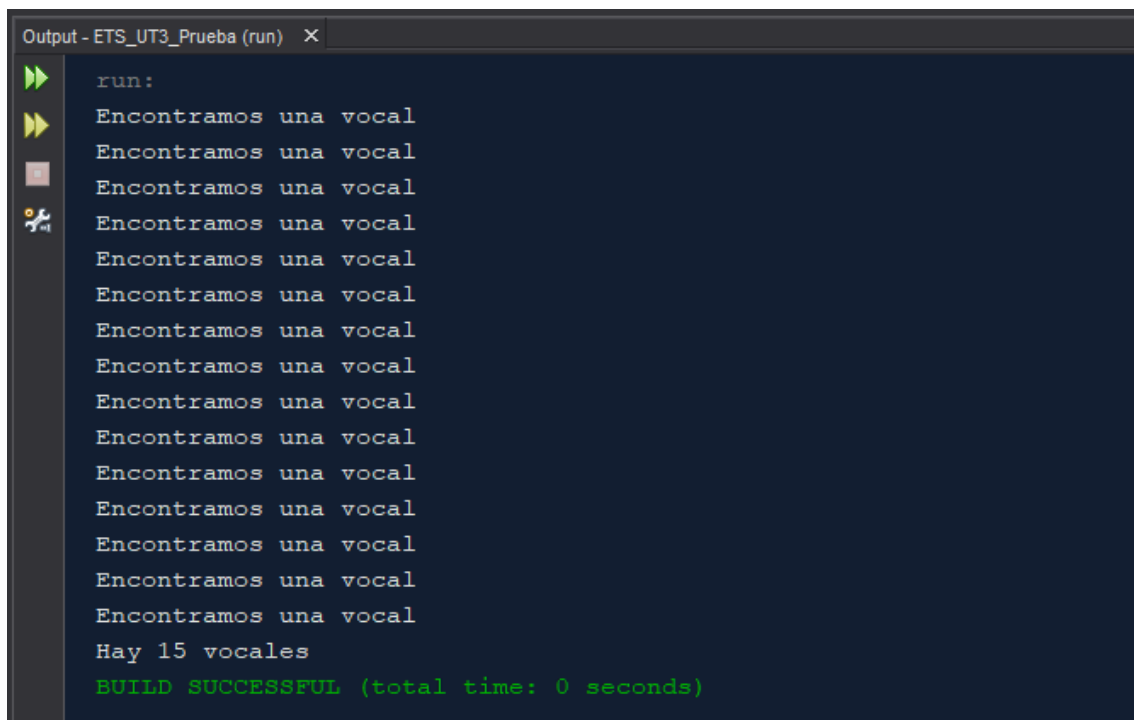
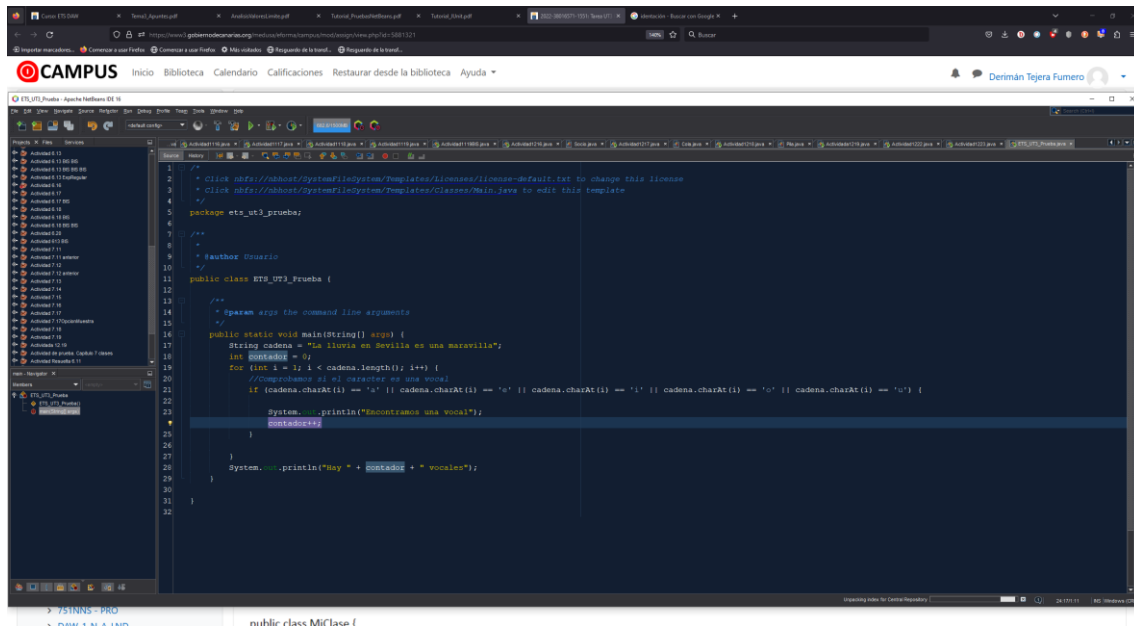
Después de pegar el código en NetBeans aparecerán los errores marcados en rojo, tal y como he modificado el IDE para que los muestre, en un color rojo muy llamativo:



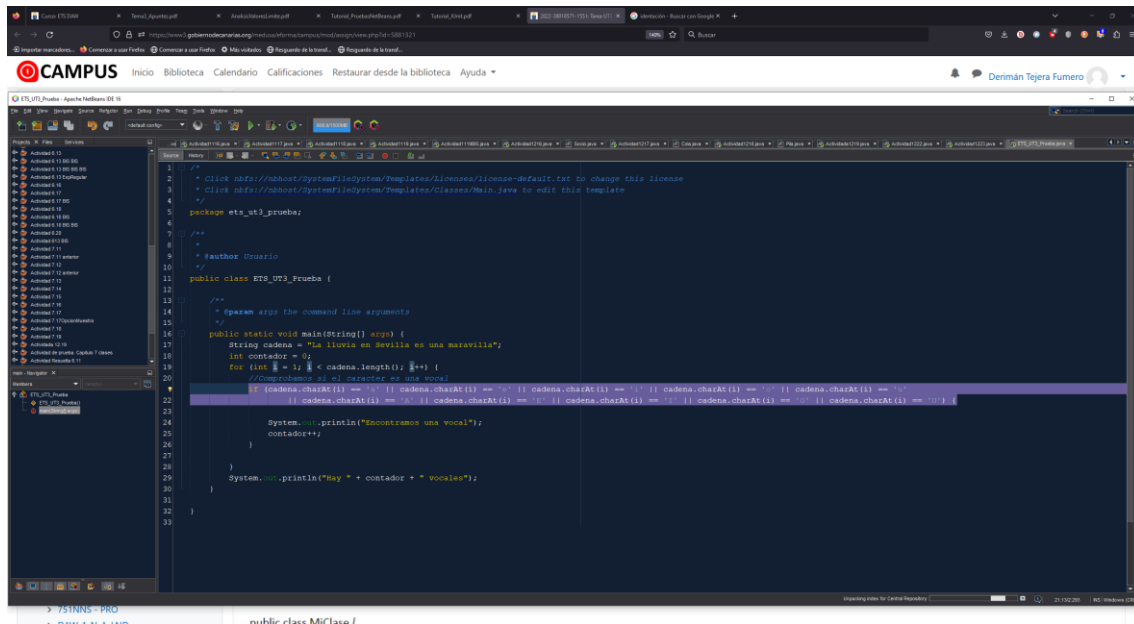
También he configurado NetBeans para que al guardar se idente el código automáticamente, al hacerlo el anterior código quedaría así:



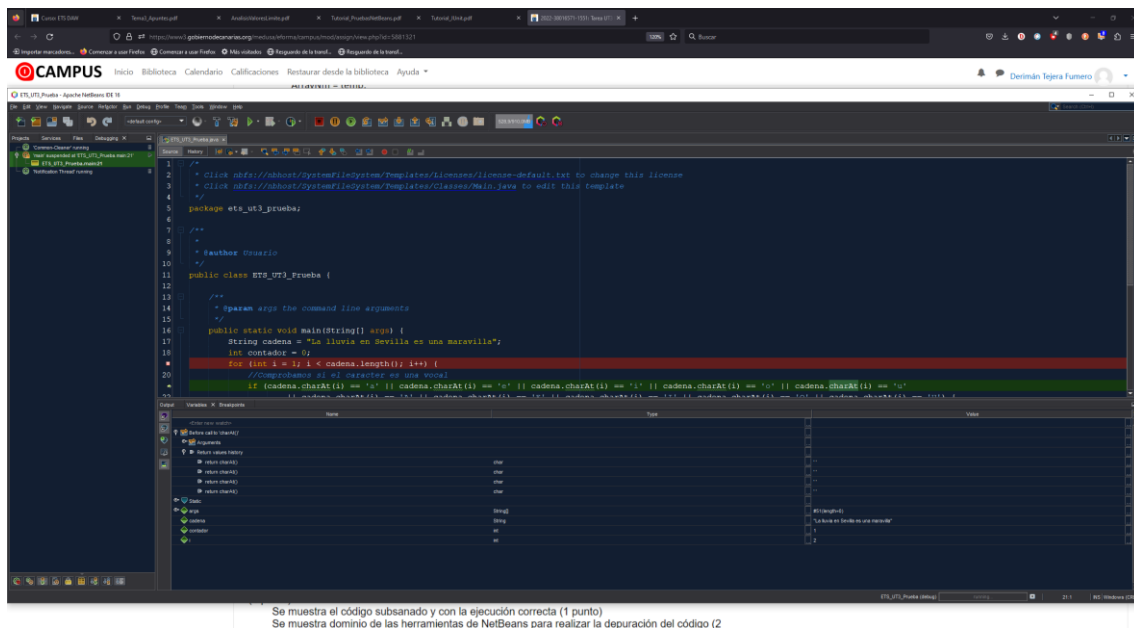
Después de resolver los errores de compilación el código quedaría así:



Ahora si muestra el número correcto de vocales, sin embargo, sólo de vocales minúsculas, que, aunque la frase de muestra no tiene vocales en mayúsculas, de usar otra podría generar problemas, por lo que se modificará el código para que cuente vocales en mayúsculas:



Estos errores fueron descubiertos gracias a usar métodos de depuración de NetBeans “debugger”, colocando breakpoints y ejecutando paso a paso, viendo cómo ejecutaba el bucle, así se vio claramente que `contador++` no estaba bien colocado en primer lugar:



Actividad 2: Elaborarás una serie de pruebas unitarias en NetBeans para aplicar diferentes test al siguiente código utilizando JUnit:

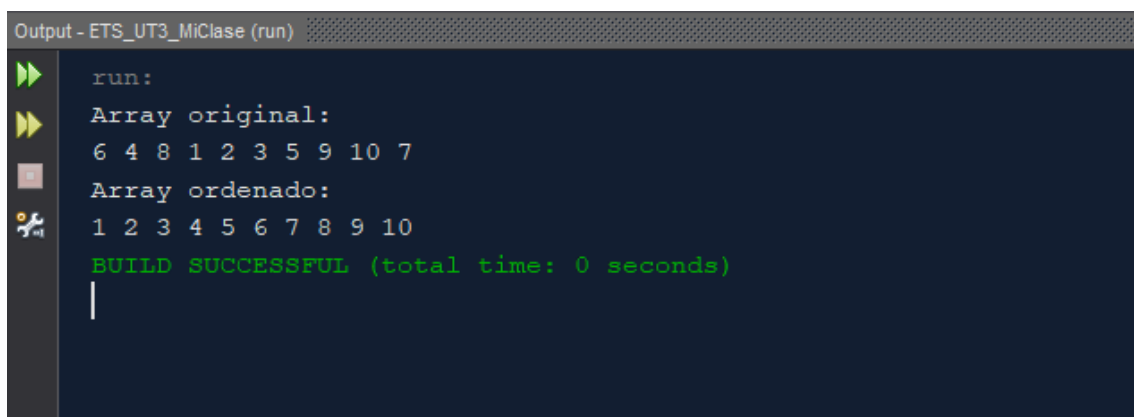

```

public class MiClase {
    public int[] burbuja(int ArrayN[]) {
        for (int i = 0; i < ArrayN.length - 1; i++) {
            for (int j = 0; j < ArrayN.length - 1; j++) {
                if (ArrayN[j] > ArrayN[j + 1]) {
                    int temp = ArrayN[j + 1];
                    ArrayN[j + 1] = ArrayN[j];
                    ArrayN[j] = temp;
                }
            }
        }
        return ArrayN;
    }
}

```

Explicación de funcionamiento del código:

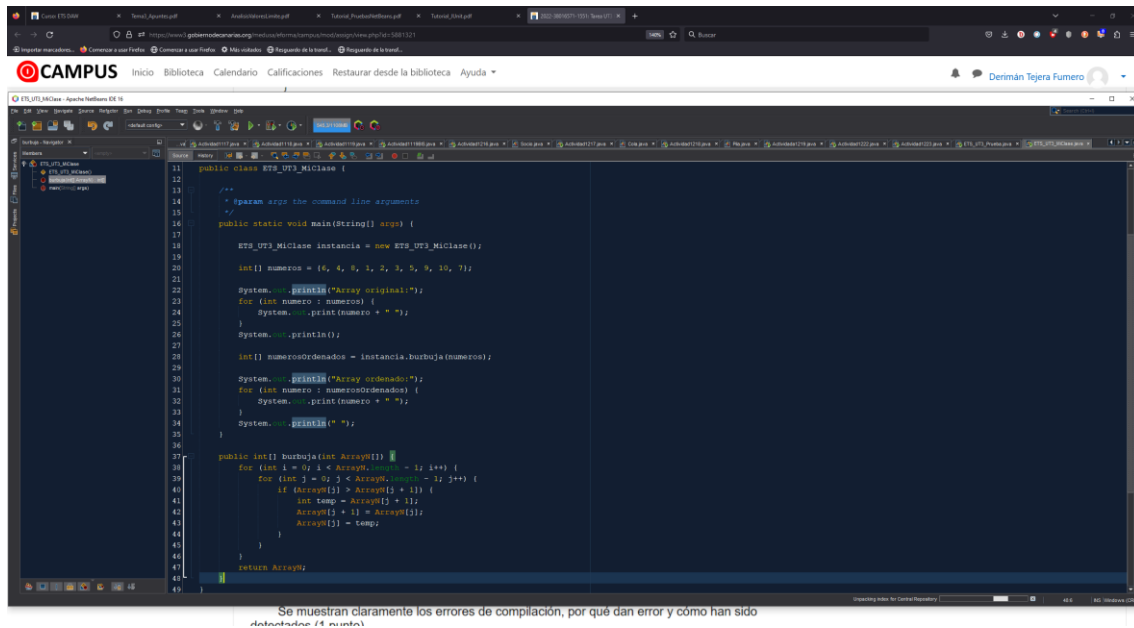
El código anterior es un algoritmo de ordenación de números, el algoritmo compara cada número con el siguiente en la tabla, desde el principio de la tabla al penúltimo, en el caso en el que el primer número lo compara con el segundo y si es este mayor, entonces lo intercambia. Todo este proceso se repite una y otra vez hasta que no se produzcan mas intercambio de posiciones entre números, entonces se entiende que los números contenidos en el array han sido completamente ordenados de menor a mayor.



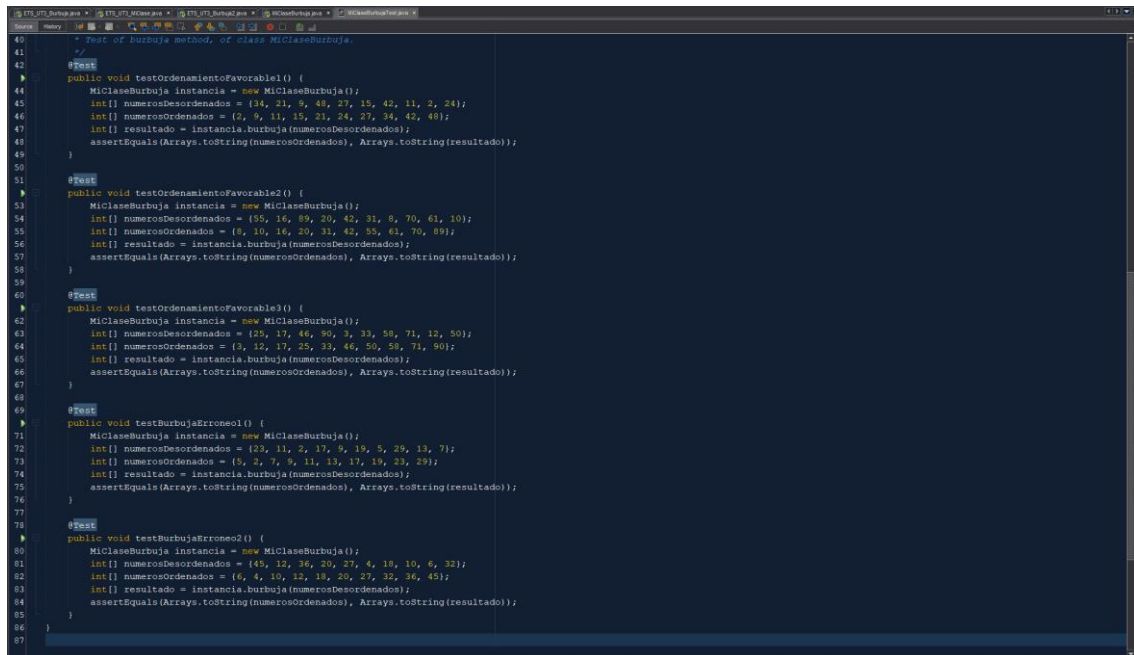
```

Output - ETS_UT3_MiClase (run)
run:
Array original:
6 4 8 1 2 3 5 9 10 7
Array ordenado:
1 2 3 4 5 6 7 8 9 10
BUILD SUCCESSFUL (total time: 0 seconds)

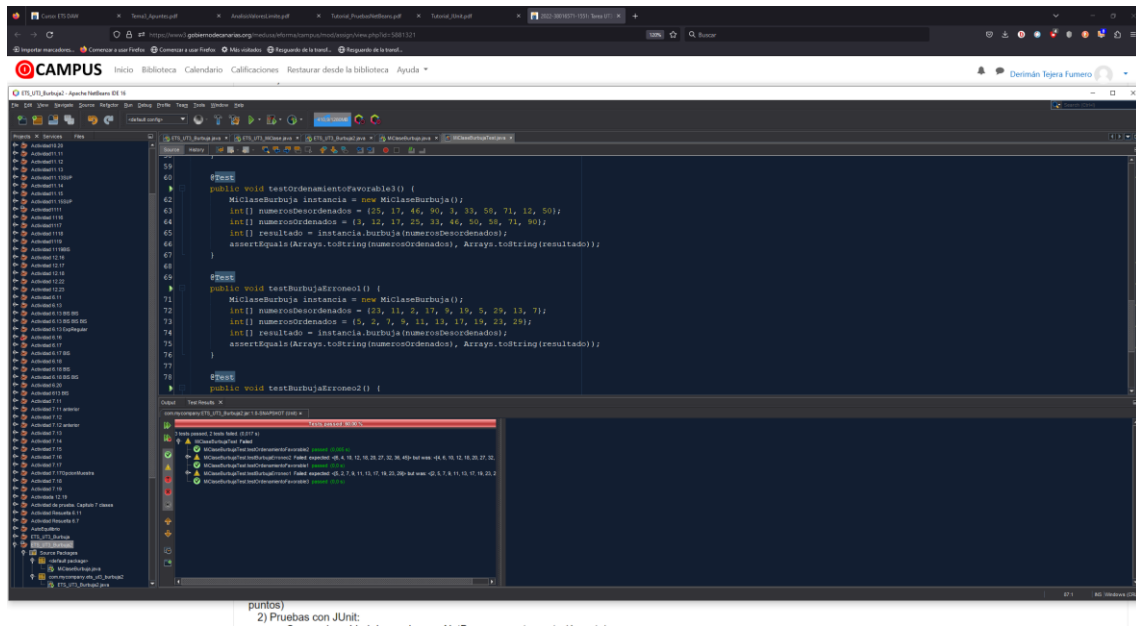
```



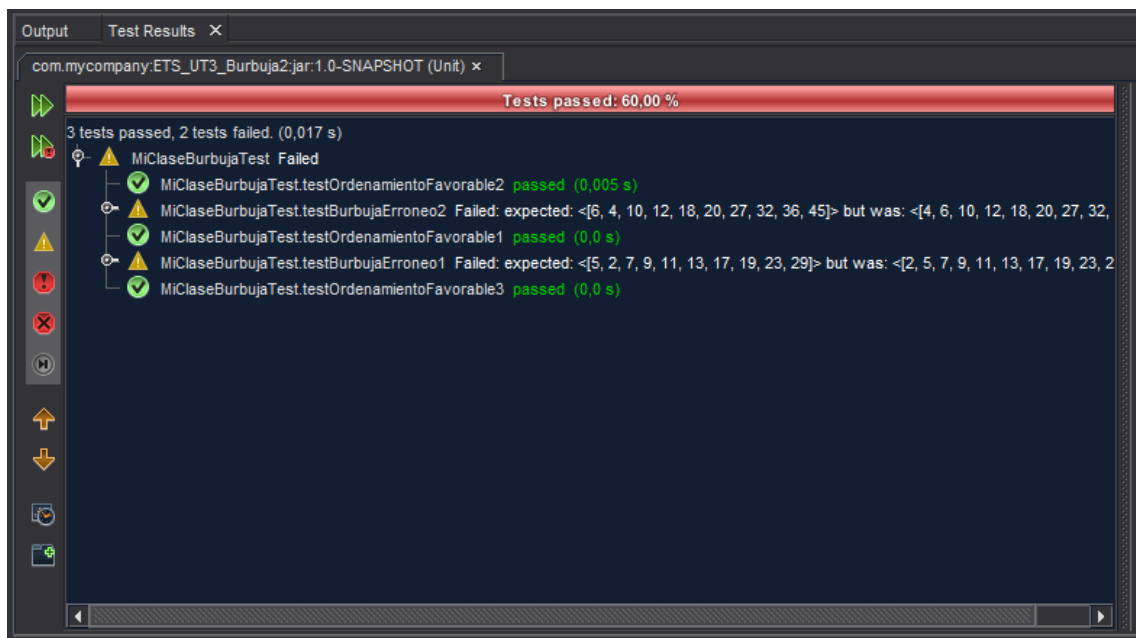
Los test que he creado son:



El resultado de los 5 test, 3 favorables y 2 erróneos es:

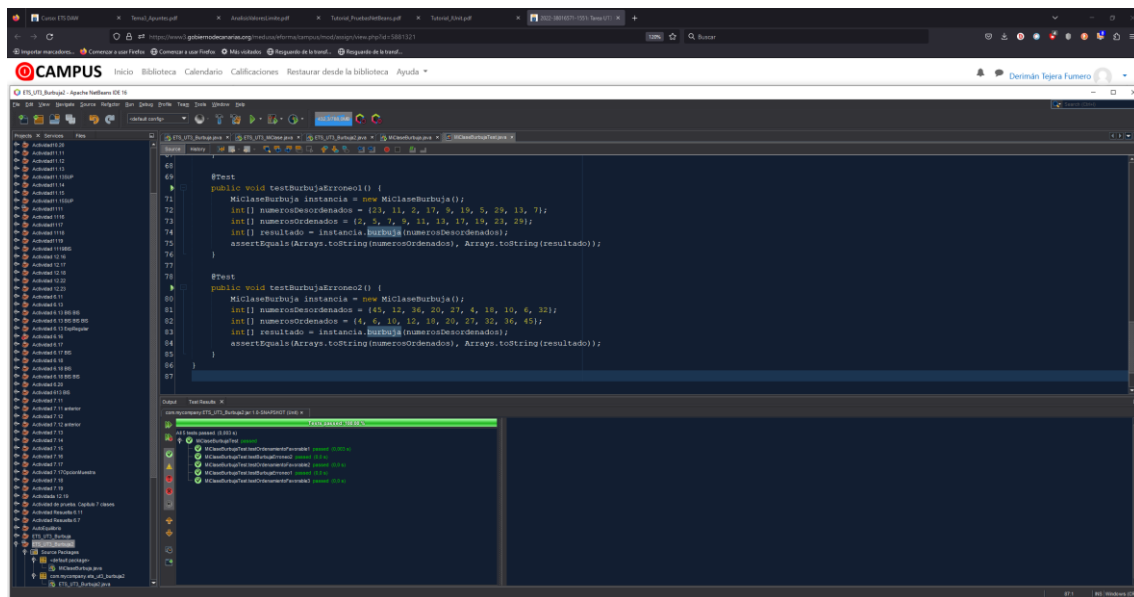


puntos)
2) Pruebas con JUnit:

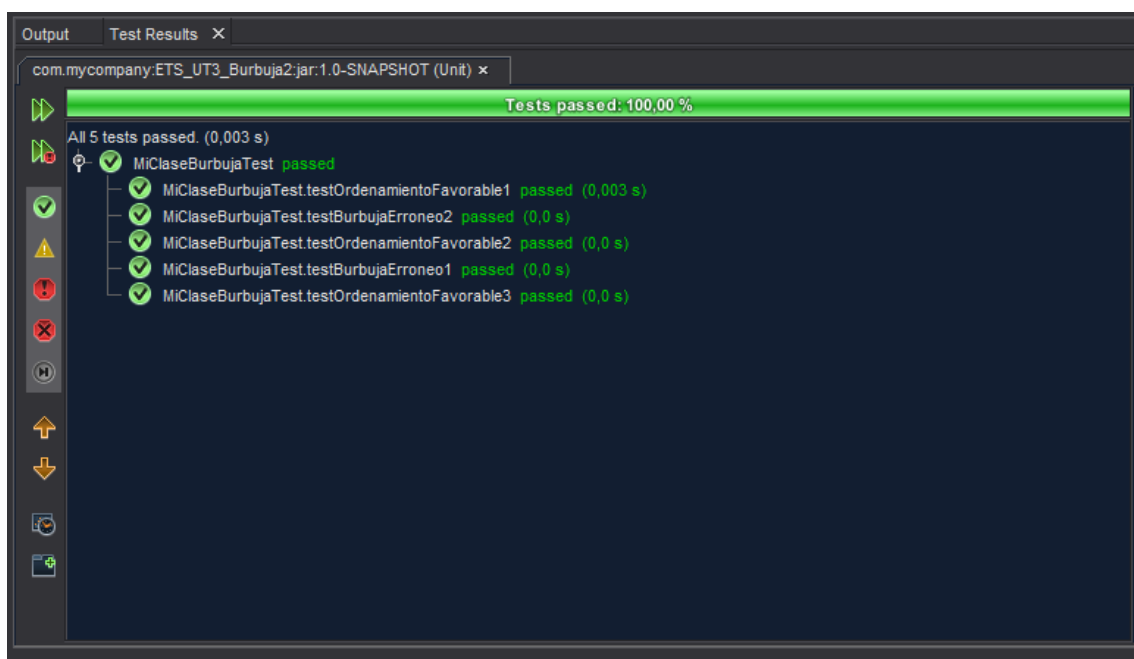


Anteriormente, para provocar los 2 test erróneos, simplemente se ha cambiado el orden de los dos primeros resultados esperados para que no coincida.

Modificando los test erróneos para que sean correctos y muestren ese resultado en el test:



Se ejecutan las pruebas y se muestran los resultados (1 punto)
 Se modifican las pruebas erróneas y se ejecutan favorablemente (1 punto)
 Esta actividad se corresponde con el resultado de aprendizaje (RA) 3: **Verifica el funcionamiento**



FIN