

CIFP César Manrique.

Entornos de desarrollo 1º de Desarrollo de Aplicaciones Web

Profesora: Sofía del Carmen Hernández González

Depuración de errores y pruebas.

Juan Carlos Francisco Mesa



Esta obra está licenciada bajo la Licencia Creative Commons Atribución 4.0 Internacional.
Para ver una copia de esta licencia, visite <http://creativecommons.org/licenses/by/4.0/> o
envíe una carta a Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Índice

Actividad 1.....	1
Enunciado.....	1
Creación del proyecto en el directorio correspondiente.....	2
Inicio del proyecto.....	4
Problemas de compilación.....	5
Problemas de ejecución.....	7
Actividad 2.....	10
Enunciado.....	10
Inicio del proyecto.....	11
Funcionamiento del código.....	14
Creación de los tests.....	15
Funcionamiento de los tests.....	16

Actividad 1

Enunciado

Tarea UT3 Depuración de errores y Pruebas

Crea un archivo en algún procesador de textos, para que puedas **añadir, paso a paso, las capturas, comentarios y respuestas** a las dos actividades propuestas a continuación:

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 **corrígelas**. 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.

```
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");  
    }  
}
```

Imagen 1: Enunciado de la actividad

[Volver al índice](#)

Creación del proyecto en el directorio correspondiente

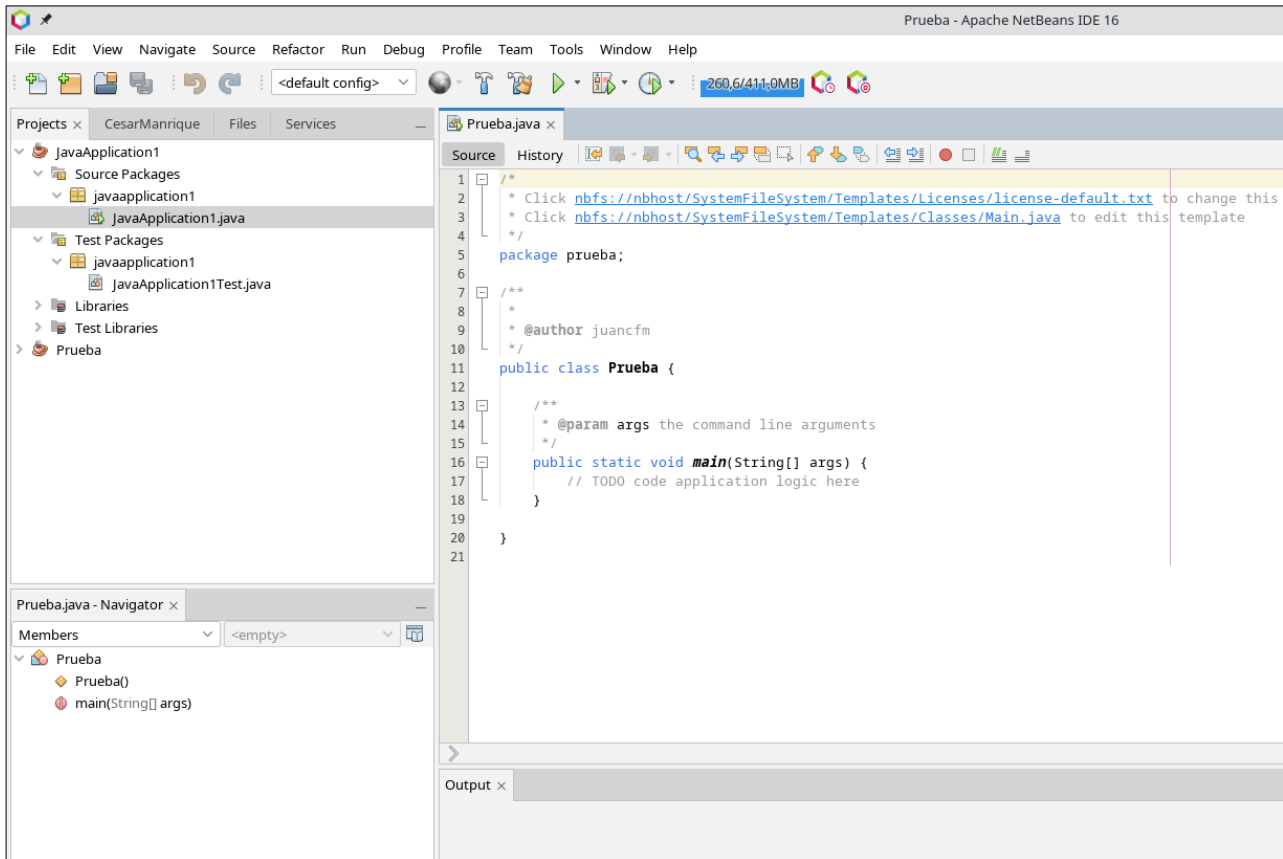


Imagen 2: Creación del proyecto, clase Prueba y método main

[Volver al índice](#)

Personalizaciones

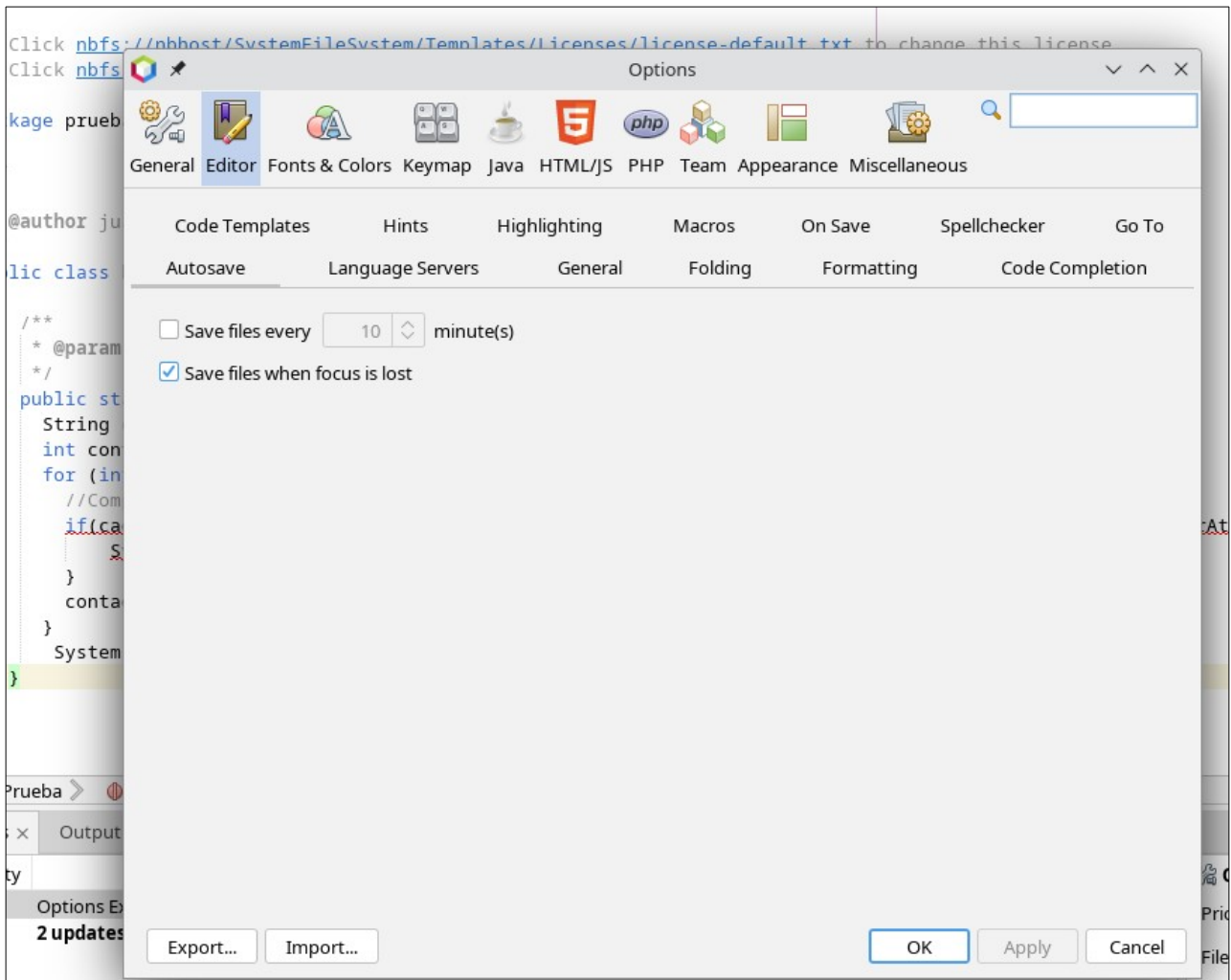


Imagen 3: Modificando las opciones

[Volver al índice](#)

Inicio del proyecto

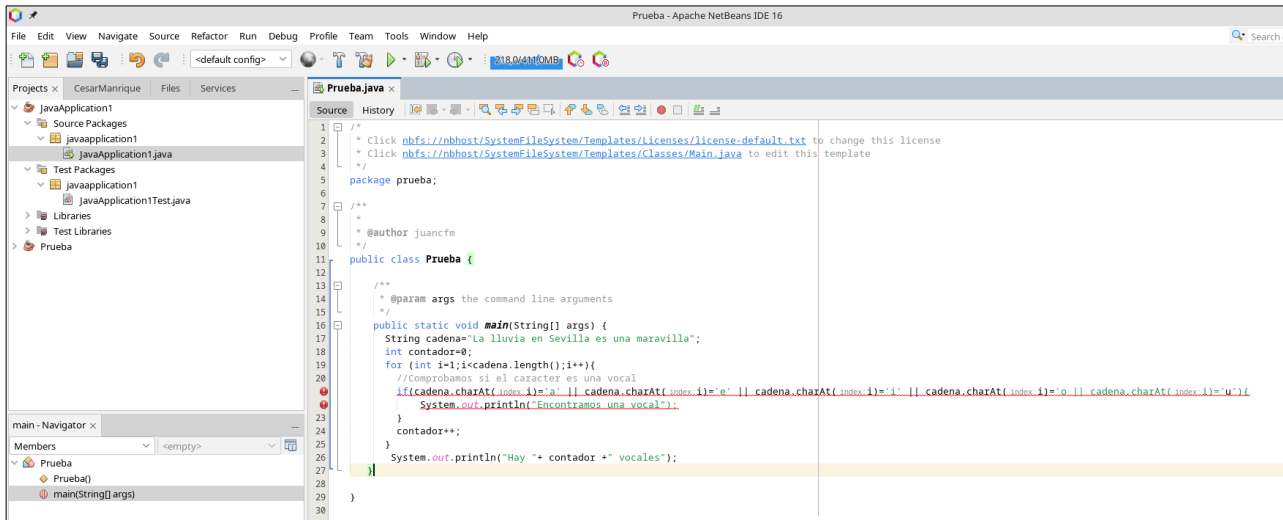


Imagen 4: Pegamos el contenido que nos fue suministrado de la clase Prueba

Problemas de compilación

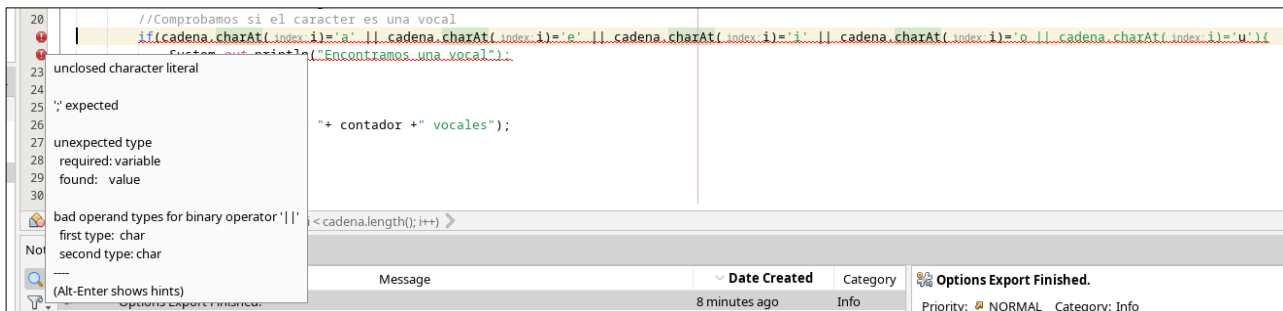


Imagen 5: En esta imagen podemos observar como el IDE nos va indicando los errores de sintaxis que va consiguiendo y los vamos corrigiendo

[Volver al índice](#)

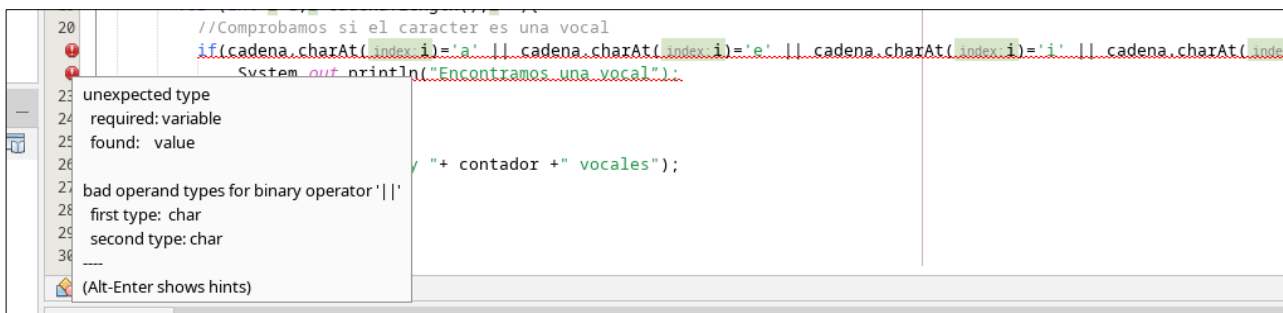


Imagen 6: En este caso vemos como nos señala errores que harán que no compile la aplicación

[Volver al índice](#)

Depuración de errores y pruebas.

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this
 */
package prueba;

/**
 *
 * @author juancfm
 */
public class Prueba {

    /**
     * @param args the command line arguments
     */
    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(index:i)=='a' ||
                cadena.charAt(index:i)=='e' ||
                cadena.charAt(index:i)=='i' ||
                cadena.charAt(index:i)=='o' ||
                cadena.charAt(index:i)=='u'){

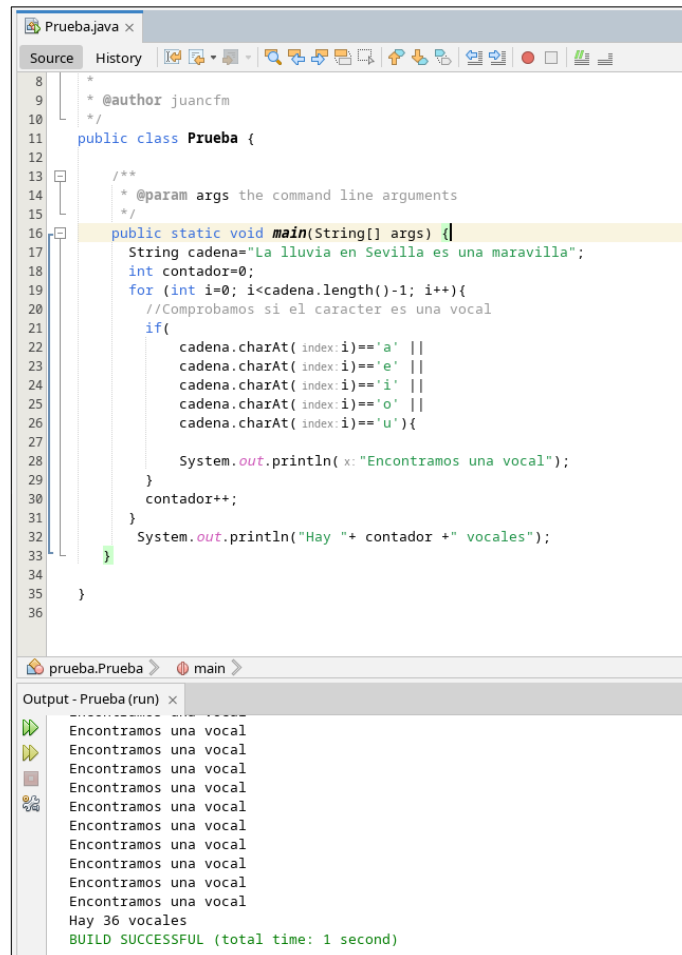
                System.out.println( x: "Encontramos una vocal");
            }
            contador++;
        }
        System.out.println("Hay "+ contador +" vocales");
    }
}

```

Imagen 7: Aquí el código una vez solucionados los errores mostrados

[Volver al índice](#)

Problemas de ejecución



```
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36

```

```

 * @author juancfm
 */
public class Prueba {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        String cadena="La lluvia en Sevilla es una maravilla";
        int contador=0;
        for (int i=0; i<cadena.length()-1; i++){
            //Comprobamos si el caracter es una vocal
            if(
                cadena.charAt( index: i)=='a' ||
                cadena.charAt( index: i)=='e' ||
                cadena.charAt( index: i)=='i' ||
                cadena.charAt( index: i)=='o' ||
                cadena.charAt( index: i)=='u' ){

                System.out.println( x: "Encontramos una vocal");
            }
            contador++;
        }
        System.out.println("Hay " + contador + " vocales");
    }
}

```

prueba.Prueba main

Output - Prueba (run)

```

Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Hay 36 vocales
BUILD SUCCESSFUL (total time: 1 second)

```

Imagen 8: Aquí podemos observar que a pesar de haber compilado correctamente, el resultado de vocales contadas no se corresponde con las que hay realmente en la frase, esto es por la mala ubicación de contador, que debe estar dentro del condicional para contar bien las vocales

[Volver al índice](#)

Depuración de errores y pruebas.

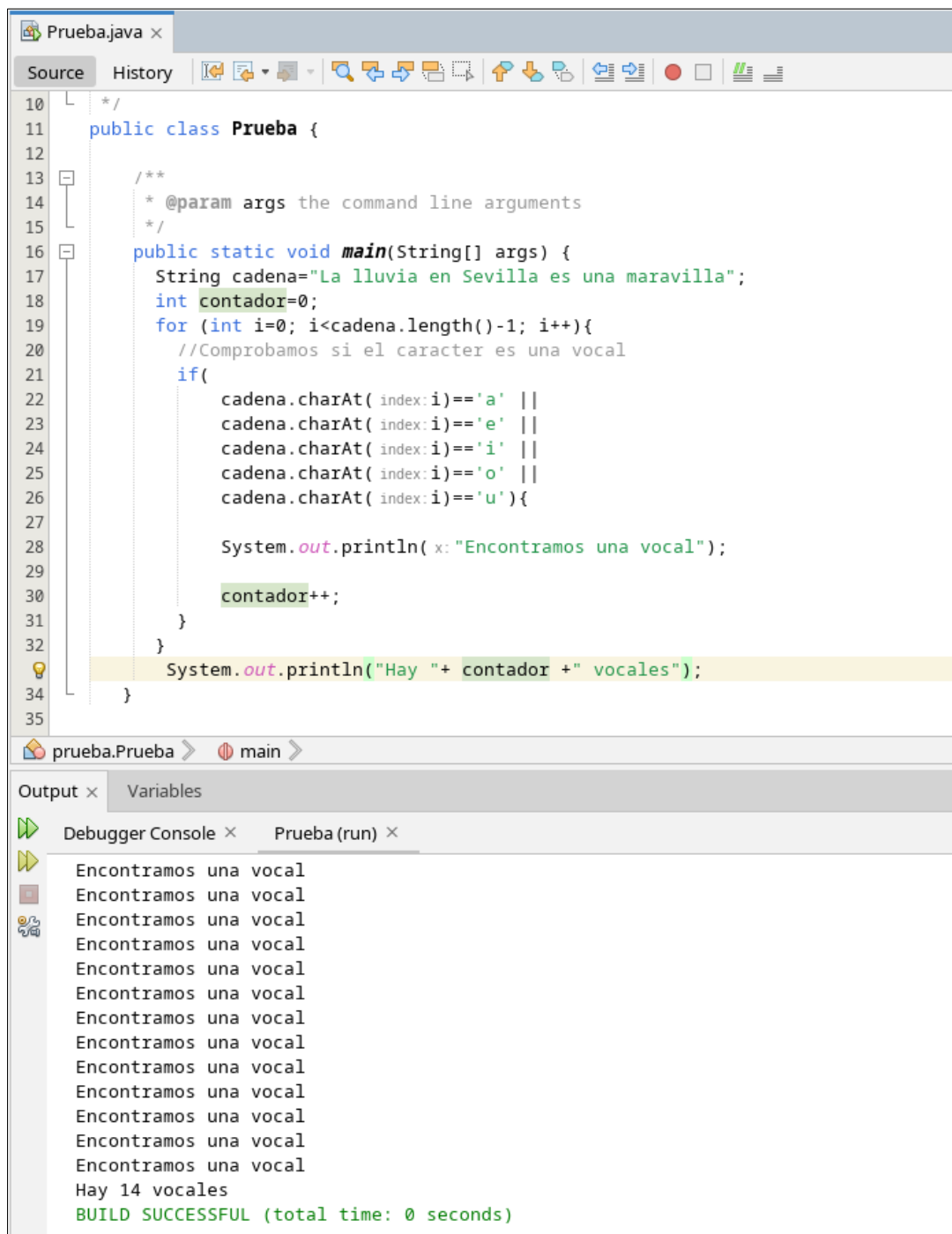
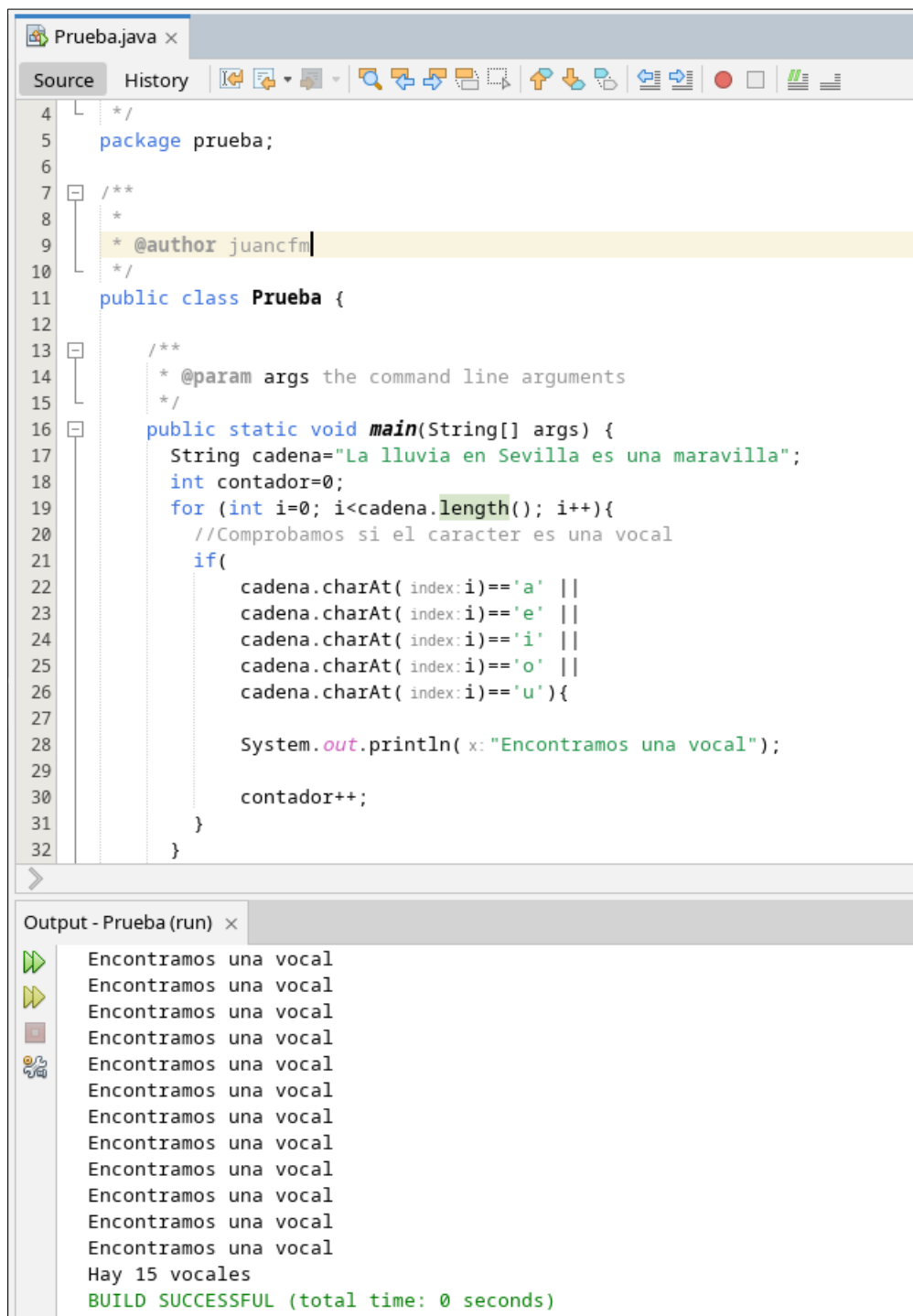


Imagen 9: En este caso tampoco está haciendo bien el conteo de vocales, pero esta vez el error se debe a la condición en el bucle que no llegará a recorrerlo completamente

[Volver al índice](#)

Depuración de errores y pruebas.



The screenshot shows an IDE window titled 'Prueba.java'. The code defines a package 'prueba' and a class 'Prueba' with a static method 'main'. The method iterates through the string 'La lluvia en Sevilla es una maravilla', checking each character for being a vowel (a, e, i, o, u). The output window shows 15 occurrences of 'Encontramos una vocal' and a final message 'Hay 15 vocales'. The build status is 'BUILD SUCCESSFUL (total time: 0 seconds)'.

```
4  */
5  package prueba;
6
7  /**
8   *
9   * @author juancfm
10  */
11  public class Prueba {
12
13      /**
14       * @param args the command line arguments
15       */
16      public static void main(String[] args) {
17          String cadena="La lluvia en Sevilla es una maravilla";
18          int contador=0;
19          for (int i=0; i<cadena.length(); i++){
20              //Comprobamos si el caracter es una vocal
21              if(
22                  cadena.charAt( index:i)=='a' ||
23                  cadena.charAt( index:i)=='e' ||
24                  cadena.charAt( index:i)=='i' ||
25                  cadena.charAt( index:i)=='o' ||
26                  cadena.charAt( index:i)=='u'){
27
28                  System.out.println( x: "Encontramos una vocal");
29
30                  contador++;
31              }
32          }
33      }
```

Output - Prueba (run) x

```
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Encontramos una vocal
Hay 15 vocales
BUILD SUCCESSFUL (total time: 0 seconds)
```

Imagen 10: Aquí podemos observar el código corregido y haciendo el conteo adecuadamente

[Volver al índice](#)

Actividad 2

Enunciado

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;  
    }  
}
```

Examina el código y **explica que hace**

Elabora un **conjunto de pruebas unitarias** que incluya 5 tests diferentes: **3** de ellos con **resultado favorable** y **2** con **resultado erróneo**. Es decir:

- **Tres de esos tests** le pasarán a la instancia de nuestra clase **un array desordenado** y tendrán como **resultado esperado** dicho **array correctamente ordenado**, por lo que nuestro programa deberá superar dichos test.
- Por otro lado, **dos de los test** le pasarán a la instancia de nuestra clase **un array desordenado** y tendrán como resultado esperado **el array introducido pero mal ordenado**, por lo que nuestro programa no deberá superar dicho test.
- También **deberás comprobar**, una vez finalizados los 5 tests, que si pones como resultado esperado el **array correctamente ordenado en los dos test desfavorables** o erróneos, nuestro programa **pasaría los cinco tests** sin problemas

Recuerda **personalizar el entorno y mostrar paso a paso todo lo que vas realizando para hacer las pruebas en JUnit**. Muestra **Las 5 pruebas** o test que utilizarás, es decir **Datos de entrada y Resultado esperado**. Realiza capturas para **poder seguir TODO el proceso** y ver **claramente los resultados**.

NOTA: Utiliza la instrucción:

```
assertEquals(Arrays.toString(expResult), Arrays.toString(result));  
Para que JUnit compare dos arrays
```

Imagen 11: Enunciado de la actividad 2

[Volver al índice](#)

Inicio del proyecto

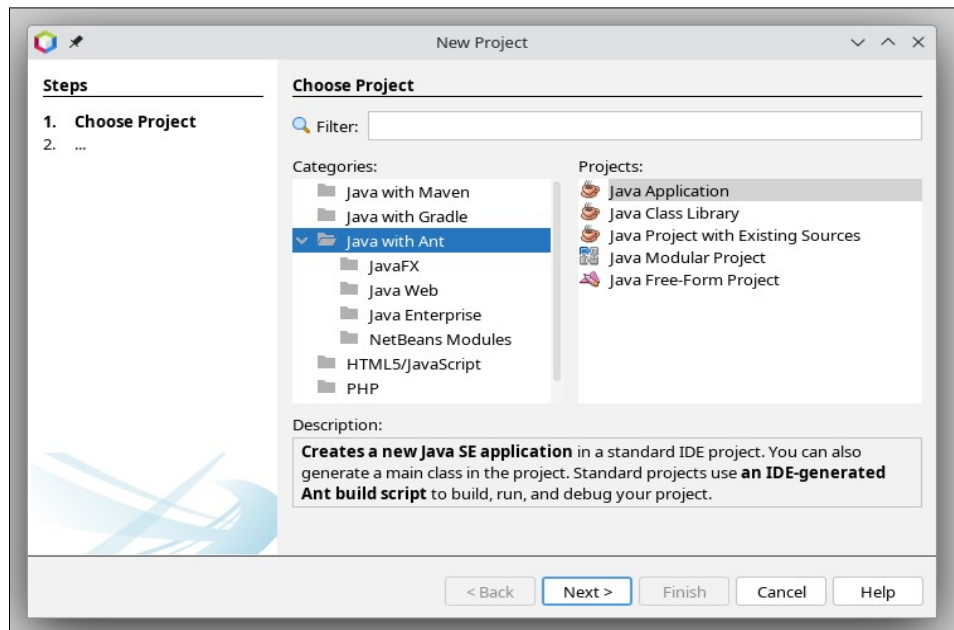


Imagen 12: Creación del proyecto

[Volver al índice](#)

Depuración de errores y pruebas.

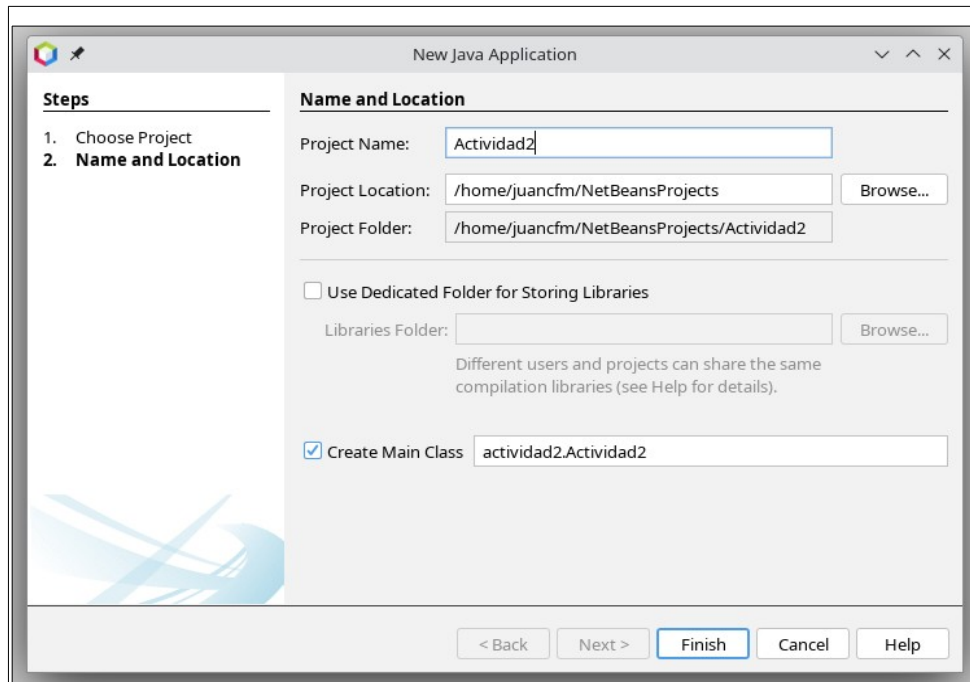


Imagen 13: Nombramos el proyecto

[Volver al índice](#)

Depuración de errores y pruebas.

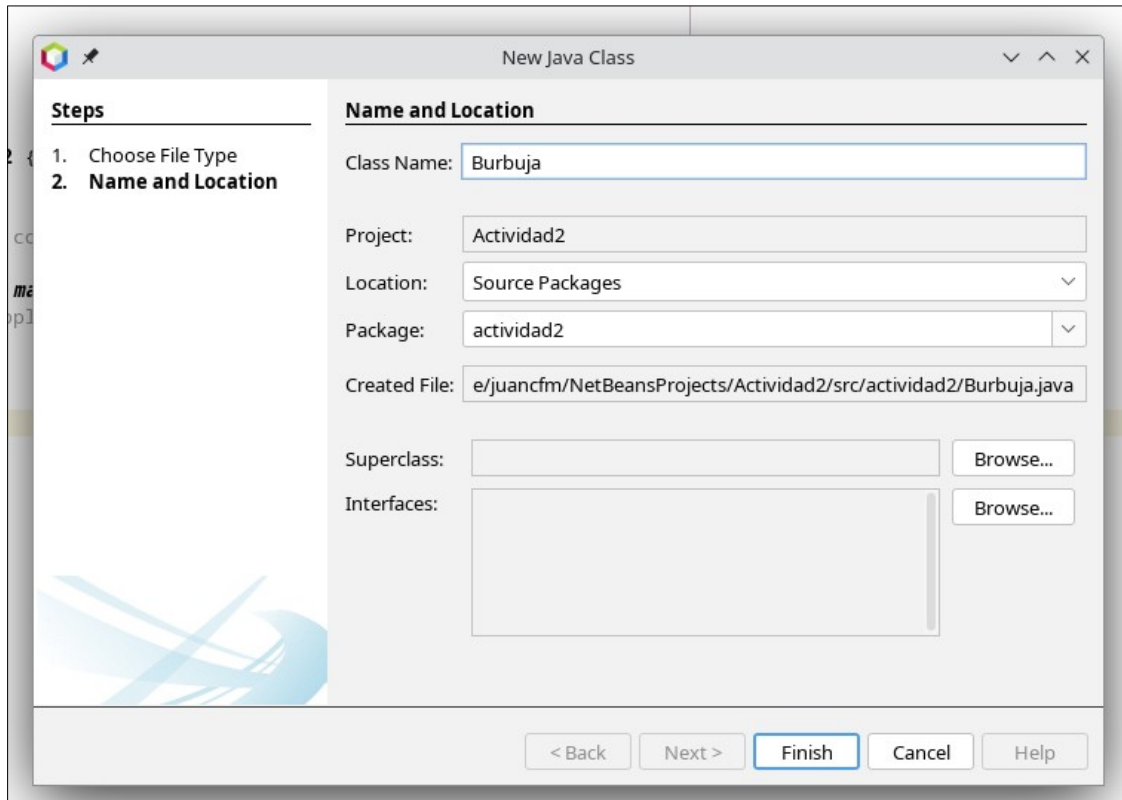


Imagen 14: Creacion de la clase Burbuja

[Volver al índice](#)

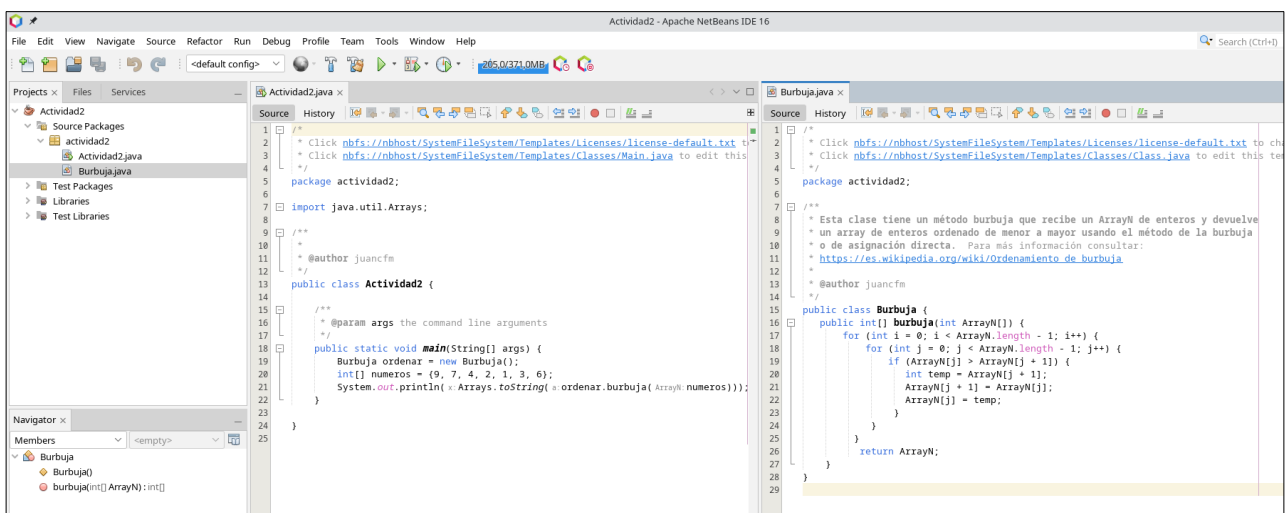


Imagen 15: Aquí podemos observar el proyecto con las dos clases creadas

[Volver al índice](#)

Funcionamiento del código

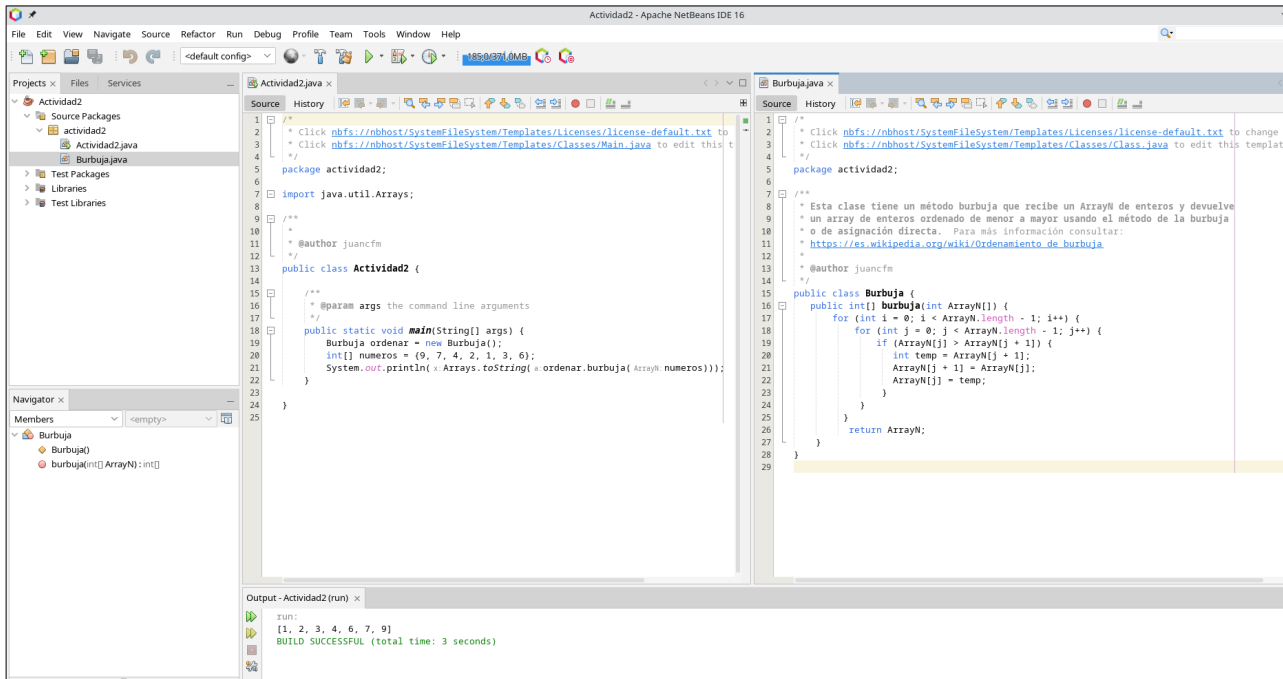


Imagen 16: Hacemos una instancia de la clase Burbuja en main y le pasamos un array de enteros. Así vemos como nos devuelve el array ordenado

[Volver al índice](#)

Creación de los tests

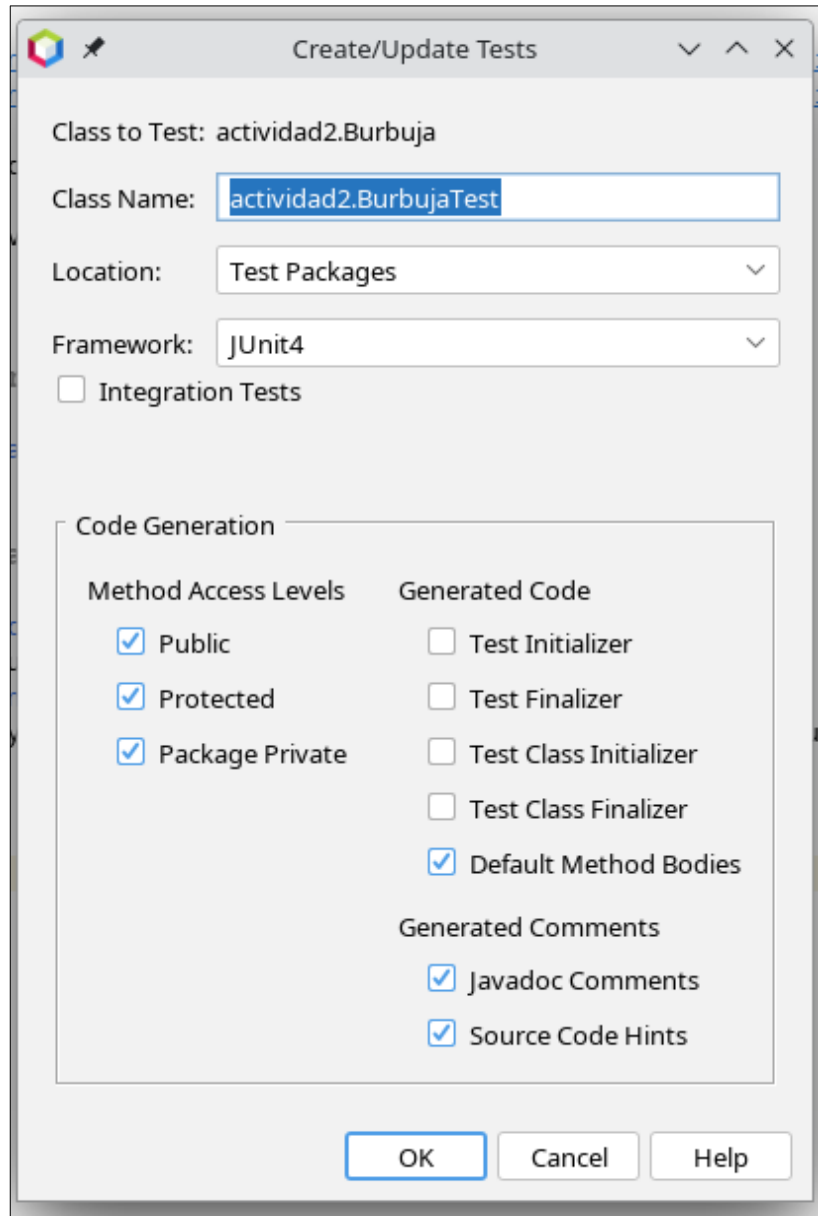


Imagen 17: Creamos los test de la clase burbuja usando desde Tools>Create/Update Tests

[Volver al índice](#)

Funcionamiento de los tests

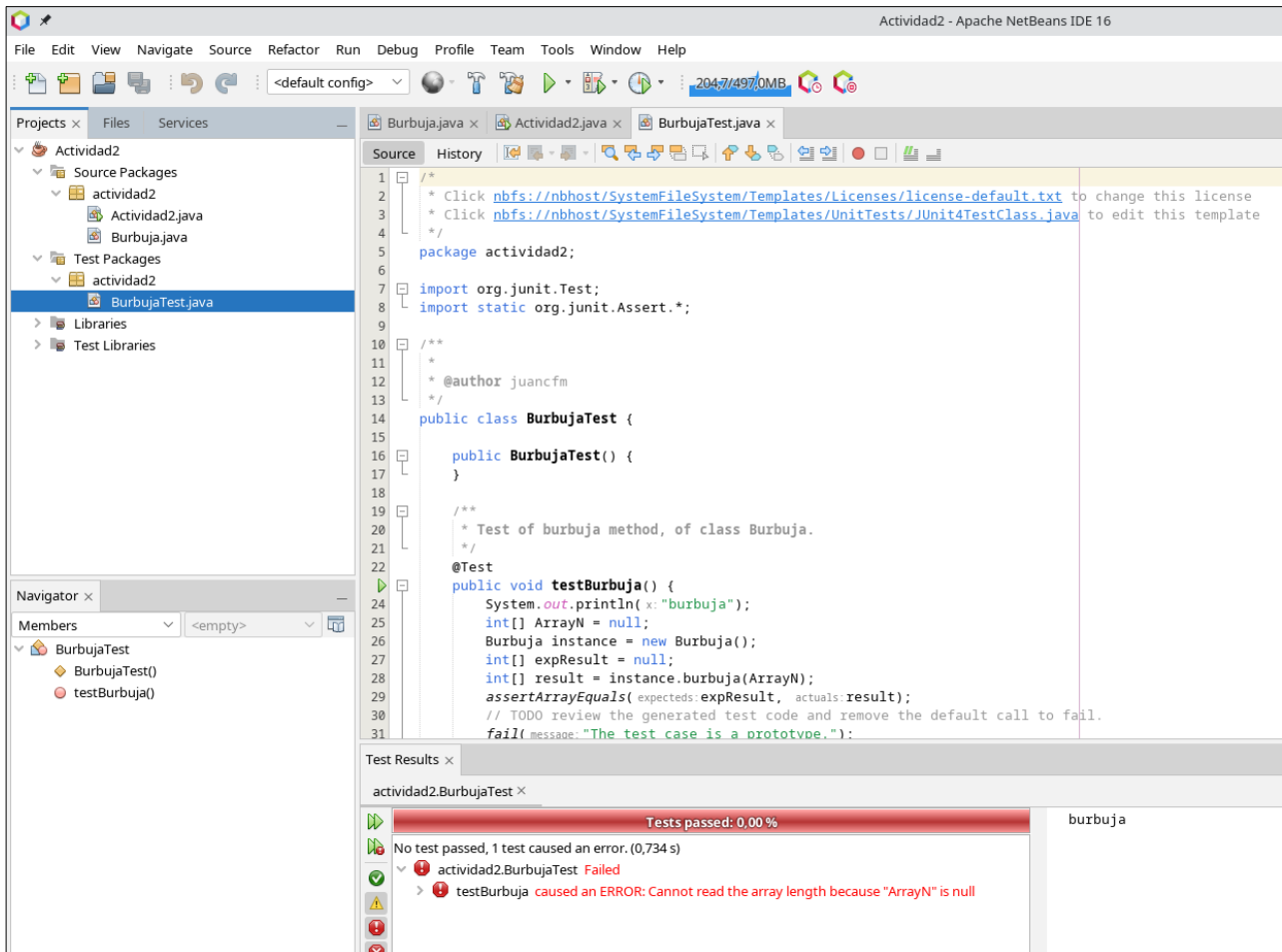


Imagen 18: Vemos como al correr el test la primera vez este falla y nos dice la causa con lo cual podremos ir arreglando lo que ha fallado

[Volver al índice](#)

Depuración de errores y pruebas.

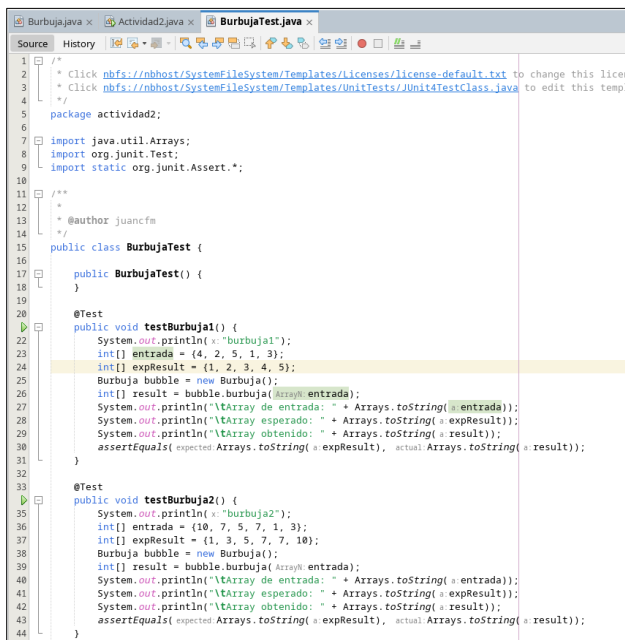


Imagen 19: Creamos los Tests.

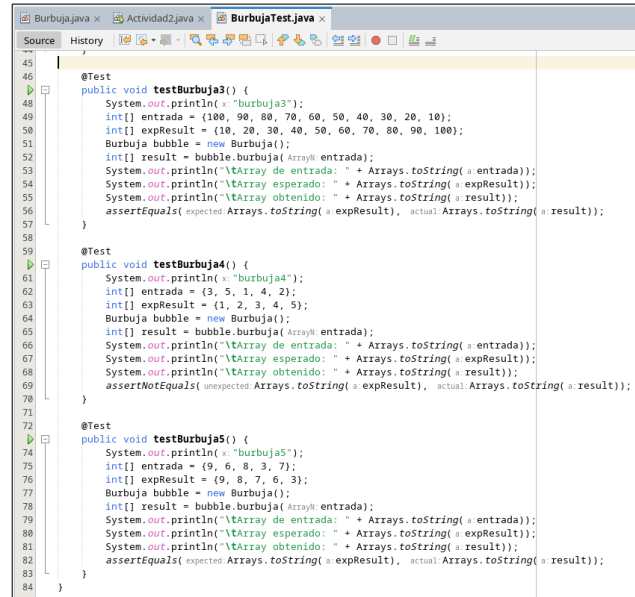


Imagen 20: Aquí podemos ver los cinco tests ya creados

[Volver al índice](#)

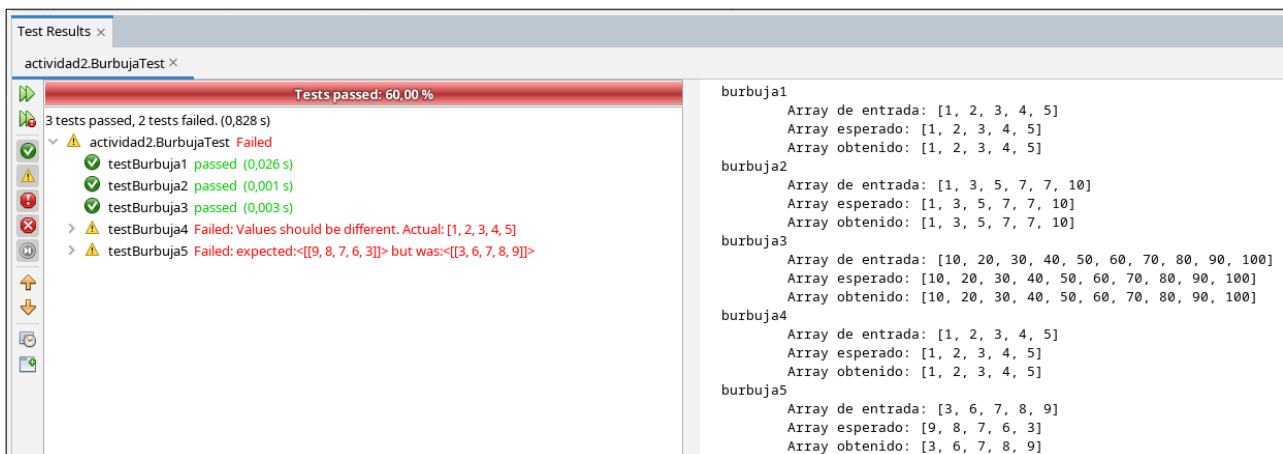


Imagen 21: Vemos como pasan los tres primeros tests mientras los dos últimos fallan tal como se pidió en el enunciado de la actividad

[Volver al índice](#)