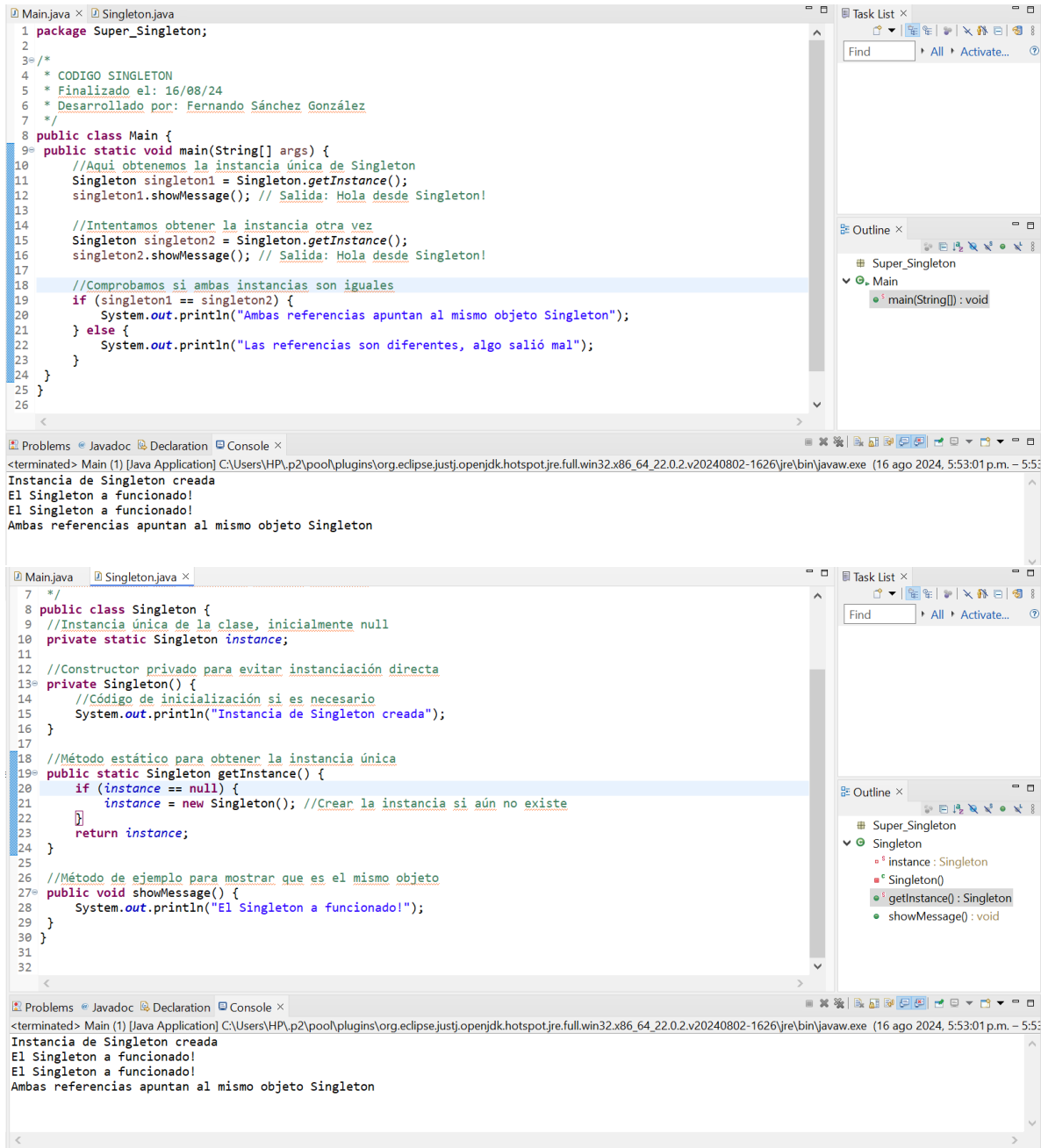


# Concepto de Singleton:

Ahora en este código de Java de Singleton, la instancia única se almacenará en una variable privada, el constructor será privado, y el acceso a la instancia se proporcionará a través del método estático `getInstance()`.



```
1 package Super_Singleton;
2
3 /*
4  * CODIGO SINGLETON
5  * Finalizado el: 16/08/24
6  * Desarrollado por: Fernando Sánchez González
7  */
8 public class Main {
9     public static void main(String[] args) {
10         //Aquí obtenemos la instancia única de Singleton
11         Singleton singleton1 = Singleton.getInstance();
12         singleton1.showMessage(); // Salida: Hola desde Singleton!
13
14         //Intentamos obtener la instancia otra vez
15         Singleton singleton2 = Singleton.getInstance();
16         singleton2.showMessage(); // Salida: Hola desde Singleton!
17
18         //Comprobamos si ambas instancias son iguales
19         if (singleton1 == singleton2) {
20             System.out.println("Ambas referencias apuntan al mismo objeto Singleton");
21         } else {
22             System.out.println("Las referencias son diferentes, algo salió mal");
23         }
24     }
25 }
26
```

```
7 /*
8  * public class Singleton {
9  * //Instancia única de la clase, inicialmente null
10  * private static Singleton instance;
11  *
12  * //Constructor privado para evitar instanciación directa
13  * private Singleton() {
14  * //Código de inicialización si es necesario
15  * System.out.println("Instancia de Singleton creada");
16  * }
17  *
18  * //Método estático para obtener la instancia única
19  * public static Singleton getInstance() {
20  * if (instance == null) {
21  * instance = new Singleton(); //Crear la instancia si aún no existe
22  * }
23  * return instance;
24  * }
25  *
26  * //Método de ejemplo para mostrar que es el mismo objeto
27  * public void showMessage() {
28  * System.out.println("El Singleton a funcionado!");
29  * }
30  * }
31  *
32  *
```

Problems Javadoc Declaration Console  
<terminated> Main (1) [Java Application] C:\Users\HP\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802-1626\jre\bin\javaw.exe (16 ago 2024, 5:53:01 p.m. - 5:53:01 p.m.)  
Instancia de Singleton creada  
El Singleton a funcionado!  
El Singleton a funcionado!  
Ambas referencias apuntan al mismo objeto Singleton

Task List  
Find All Activate...  
Outline  
Super\_Singleton  
Main  
main(String[]): void

Problems Javadoc Declaration Console  
<terminated> Main (1) [Java Application] C:\Users\HP\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_22.0.2.v20240802-1626\jre\bin\javaw.exe (16 ago 2024, 5:53:01 p.m. - 5:53:01 p.m.)  
Instancia de Singleton creada  
El Singleton a funcionado!  
El Singleton a funcionado!  
Ambas referencias apuntan al mismo objeto Singleton

Task List  
Find All Activate...  
Outline  
Super\_Singleton  
Singleton  
instance : Singleton  
Singleton()  
getInstance(): Singleton  
showMessage(): void

## Explicando como funcionan los .java en el código del Singleton

1. **Singleton.java:** Esta clase implementa el patrón Singleton.
  - **instance:** Una variable estática privada que mantiene la única instancia de la clase.
  - **getInstance():** Un método estático que devuelve la instancia única, creando una nueva solo si no existe.
  - **showMessage():** Un método de ejemplo para demostrar que es el mismo objeto.
2. **Main.java:** Aquí es donde probamos el patrón Singleton.
  - **singleton1 y singleton2:** Ambas referencias apuntan a la misma instancia, lo que se comprueba al final.