Documentación código Counter

Rodrigo Martínez Zambrano

El patrón de diseño Singleton está diseñado para restringir la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto. Su objetivo es garantizar que una clase solo tenga una instancia y proporcionar un punto de acceso global a esta.

En el siguiente ejemplo se realiza un código muy sencillo sobre un contador y su manipulación utilizando el patrón Singleton.

Se tiene la clase Counter.java en la cual se indica lo siguiente:

Se crean los atributos privados y estáticos para mantener encapsulación y mantener su valor en memoria.

```
// La única instancia de esta clase
private static Counter counterIns;  3 usages
private static int count;  3 usages
```

Posteriormente el acceso del constructor se define como privado para evitar la creación de instancias por aparte.

```
// Constructor privado para evitar la creación de instancias desde fuera de la clase
private Counter() {  1 usage
    count = 0;
}
```

En el método getInstance() se verifica si ya existe una instancia creada de este objeto, de no ser así crea una única instancia.

```
// Método estático para obtener la única instancia de la clase
public static Counter getInstance() {  2 usages
    if (counterIns == null) {
        counterIns = new Counter();
    }
    return counterIns;
}
```

El método increment() incrementa el valor del contador.
El método getCount() devuelve el valor actual del contador.

En la clase Main, se ejecuta una prueba del código, al crear los objetos se puede observar que contienen la misma referencia en memoria.

```
// Obtenemos la instancia de Counter
System.out.println("Creando la instancia...");
Counter counter1 = Counter.getInstance();
Counter counter2 = Counter.getInstance();

System.out.println(counter2);
System.out.println(counter1);
```

```
com.singleton.Counter@4554617c
com.singleton.Counter@4554617c
```

Posteriormente se puede verificar si ambas referencias sean iguales:

```
// Verificamos que ambas referencias son iguales
System.out.println("¿counter1 y counter2 son la misma instancia? " + (counter1==counter2));
```

```
¿counter1 y counter2 son la misma instancia? true
```

El uso del patrón Singleton es útil cuando se necesita una única instancia de una clase para toda la aplicación.