Xideral

Academia Java

Semana 1 -Dia 1

Singleton

Presentado por:

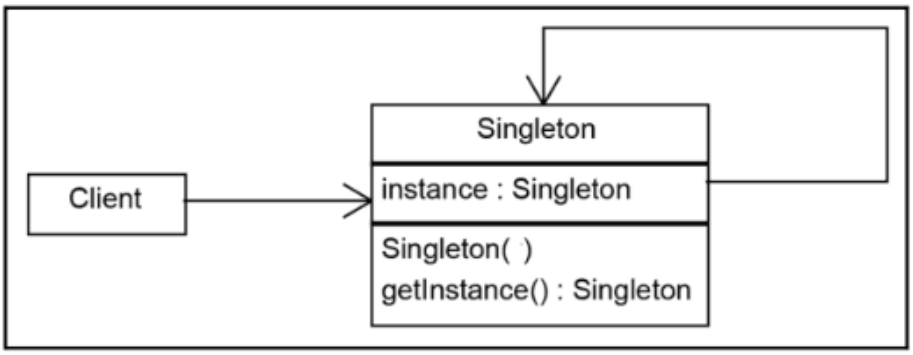
Edgar Itzak Sánchez Rogers

Monterrey Nuevo León México a 15 de agosto de 2024

Introducción:

Un singleton es un patrón de diseño que consiste en regresar la misma instancia para cada valor de referencia que hace, esto permite tener un control que toda información que envian las variables de referencia pasan por la misma instancia, sus usos más comunes son:

- Controlar el Acceso a Recursos Compartidos
- Gestión de Estado Global
- Implementación de Recursos Limitados
- Facilitación de Comunicación entre Componentes
- Instanciación Costosa



Demostración de Singleton en Java:

A continuación, se muestra un código en Java de demostración que implementa el concepto de singleton:

Lógica de singleton (Logger):

```java
package com.edgaritzak.logger;

import java.time.LocalDateTime;

public class Logger {
    private static Logger instance;
    private static String log = "";

    private Logger() {

    }
    //returns the same instance
    public static Logger getInstance() {
        if(instance == null) {
            instance = new Logger();
        }
        return instance;
    }

    //add a log to the string
    public void writeLog(String message) {
        log= log +"["+LocalDateTime.now().truncatedTo(ChronoUnit.SECONDS)+"]:"+message+"\n";
    }
    //print all logs
    public static void showLogs() {
        if(log.length()>1)
        System.out.println(log);
    }
    //clear all logs
    public static void clearLogs() {
        log="";
    }
}
```

Contiene una variable de tipo String log en la que se almacenaran todos los registros, además tiene 4 métodos: getInstance() que es que realiza el singleton al regresar la misma instancia, writeLog() agrega una línea de registros al log, showLogs() imprime todos los logs guardados, por último, clearLogs() limpia todos los logs almacenados

Clase(Main)

```java
package com.edgaritzak.logger;

public class Main {

    public static void main(String[] args) {

        //different reference variables, same instance
        Logger l1 = Logger.getInstance();
        Logger l2 = Logger.getInstance();
        Logger l3 = Logger.getInstance();

        // Call writeLog() method from different variable with the same instance. Add a log to the string
        l1.writeLog("Variable1 - Hello");
        l2.writeLog("Variable2 - Salut");
        l3.writeLog("Variable3 - Hallo");
        l1.writeLog("Variable1 - 你好");
        l2.writeLog("Variable2 - こんにちは");
        l3.writeLog("Variable3 - 안녕하세요");

        //Show the all logs
        Logger.showLogs();

        //remove all logs
        Logger.clearLogs();

        //add a new log
        l1.writeLog("Variable1 - Hola");

        //show all logs
        Logger.showLogs();
    }

}
```

Se crean 3 variables de referencia y se le asigna a la misma instancia del logger, por lo que al llamar la función de writeLog(); por diferentes variables de referencia, se agregaran registros a la misma instancia. Después de agregar registros, se imprime los registros. Para probar los otros métodos, se borran los registros, luego se agrega un registro nuevo y se vuelve a mostrar los registros

Ejecución:

```
[2024-08-15T20:42:36]:Variable1 - Hello
[2024-08-15T20:42:36]:Variable2 - Salut
[2024-08-15T20:42:36]:Variable3 - Hallo
[2024-08-15T20:42:36]:Variable1 - 你好
[2024-08-15T20:42:36]:Variable2 - こんにちは
[2024-08-15T20:42:36]:Variable3 - 안녕하세요

[2024-08-15T20:42:36]:Variable1 - Hola
```

Se muestran los logs guardados que fueron enviados por diferentes instancias de logger

Conclusión:

El patrón Singleton nos permite acceder a un objeto desde cualquier parte del programa. No obstante, también evita que otro código sobreescriba esa instancia, por lo que nos permite tener mayor seguridad y control sobre esa instancia.

Referencias:

[1] Singleton Method Design Pattern - GeeksforGeeks

[2] Singleton (refactoring.guru)