Implementación del patrón de diseño Singleton en Java

El patrón de diseño creacional **Singleton** se emplea para garantizar que una clase tenga únicamente una instancia y proporciona un punto de acceso global a esa instancia.

Este documento presenta una implementación del patrón **Singleton** en Java a través de la clase **Settings**, la cual se encarga de gestionar y almacenar la configuración global de un juego, incluyendo aspectos como calidad gráfica, resolución, volumen de sonido e idioma de texto y voces.

La elección de utilizar el patrón **Singleton** para la clase **Settings** se fundamenta en la necesidad de manejar de manera exclusiva la configuración global del juego. En el contexto de un juego, es probable que existan múltiples interfaces gráficas desde donde se accede y modifica la configuración global.

Por lo anterior, la elección del patrón **Singleton** garantiza la existencia de solo una instancia de la clase **Settings** en todo el programa, asegurando así la unicidad de la configuración del juego. Este enfoque minimiza posibles conflictos y garantiza la coherencia en todo el sistema al proporcionar un único punto de acceso global a esta instancia única permitiendo así la obtención y modificación de la configuración global desde cualquier parte del código de manera consistente.

Constants

En primer lugar, se definió la clase `Constants` que cumple con la función de almacenar las constantes relacionadas con el ejemplo. La separación de las contantes en un archivo dedicado ofrece los beneficios de mantenimiento, modularidad, organización y comprensión del código.

```
package com.game.util;

/**
 *  Clase que contiene constantes relacionadas con la configuración del juego.
 */
public class Constants {

    public static final String QUALITY_MEDIUM = "Medium";
    public static final int FPS_30 = 30;
    public static final String LANGUAGE_ENGLISH = "English";

}
```

Settings

La clase `Settings` implementa el patrón **Singleton** y muestra las variables relacionadas con las configuraciones gráficas, de sonido e idioma del juego.

En esta clase se encuentra el siguiente desglose de la implementación:

- **Instancia Singleton**. Se utiliza el campo estático `settings` y un método estático `getSettings()` para obtener la única instancia de la clase.
- **Constructor privado**. El constructor de la clase es privado para evitar instanciaciones directas desde clases externas.
- **Inicialización de configuración**. Se utiliza un método privado `initializeSettings` para establecer las configuraciones predeterminadas de gráficos, sonido e idioma al inicializar la instancia Singleton.
- **Método toString**. Se sobreescribe el método `toString` para proporcionar una representación en formato `String` de la configuración actual.

```
package com.game.config;

import static com.game.util.Constants.*;

/**
 * Implementación del patrón Singleton para gestionar y almacenar
 * la configuración de un juego, incluyendo calidad gráfica, resolución,
 * volumen de sonido y configuraciones de idioma.
 */
public class Settings {

    // Instancia Singleton.
    private static Settings settings;

    // Atributos de Configuración de Gráficos.
    private String graphicSettingsGraphicsQuality;
    private String graphicSettingsResolution;
    private int graphicSettingsFPS;

    // Atributos de Configuración de Sonido.
    private int soundSettingsMasterVolume;

    // Atributos de Configuración de Idioma.
    private String languageSettingsText;
    private String languageSettingsVoice;

    /**
```

```
     * Constructor privado para asegurar que ninguna clase externa
     * pueda instanciar Settings directamente.
     * Inicializa las configuraciones predeterminadas para gráficos,
     * sonido e idioma.
     */
    private Settings() {
        initializeSettings();
    }

    /**
     * Obtiene la instancia Singleton de Settings. Si no existe, crea
     * una nueva instancia.
     *
     * @return La instancia Singleton de Settings.
     */
    public static Settings getSettings() {
        if (settings == null)
            settings = new Settings();
        return settings;
    }

    /**
     * Devuelve una representación en formato String de la configuración actual.
     *
     * @return La cadena que representa la configuración actual de Settings.
     */
    @Override
    public String toString() {
        return "Settings {" + '\n' +
                "graphicSettingsGraphicsQuality = " + graphicSettingsGraphicsQuality + '\n' +
                "graphicSettingsResolution = " + graphicSettingsResolution + '\n' +
                "graphicSettingsFPS = " + graphicSettingsFPS + '\n' +
                "soundSettingsMasterVolume = " + soundSettingsMasterVolume + '\n' +
                "languageSettingsText = " + languageSettingsText + '\n' +
                "languageSettingsVoice = " + languageSettingsVoice + '\n' +
                '}';
    }

    // Getters y Setters para las configuraciones de gráficos, sonido e idioma.
    public String getGraphicSettingsGraphicsQuality() {
        return graphicSettingsGraphicsQuality;
    }

    public void setGraphicSettingsGraphicsQuality(String graphicSettingsGraphicsQuality) {
        this.graphicSettingsGraphicsQuality = graphicSettingsGraphicsQuality;
    }
```

```java
public String getGraphicSettingsResolution() {
    return graphicSettingsResolution;
}

public void setGraphicSettingsResolution(String graphicSettingsResolution) {
    this.graphicSettingsResolution = graphicSettingsResolution;
}

public int getGraphicSettingsFPS() {
    return graphicSettingsFPS;
}

public void setGraphicSettingsFPS(int graphicSettingsFPS) {
    this.graphicSettingsFPS = graphicSettingsFPS;
}

public int getSoundSettingsMasterVolume() {
    return soundSettingsMasterVolume;
}

public void setSoundSettingsMasterVolume(int soundSettingsMasterVolume) {
    this.soundSettingsMasterVolume = soundSettingsMasterVolume;
}

public String getLanguageSettingsText() {
    return languageSettingsText;
}

public void setLanguageSettingsText(String languageSettingsText) {
    this.languageSettingsText = languageSettingsText;
}

public String getLanguageSettingsVoice() {
    return languageSettingsVoice;
}

public void setLanguageSettingsVoice(String languageSettingsVoice) {
    this.languageSettingsVoice = languageSettingsVoice;
}

/**
 * Inicializa las configuraciones predeterminadas para gráficos,
 * sonido e idioma.
 */
private void initializeSettings() {
    setGraphicsDefaultQuality();
    initializeDefaultSoundSettings();
```

```
        initializeDefaultLanguageSettings();
    }

    /**
     * Establece la calidad gráfica, resolución y FPS predeterminados.
     */
    private void setGraphicsDefaultQuality() {
        setGraphicSettingsGraphicsQuality(QUALITY_MEDIUM);
        setGraphicSettingsResolution(QUALITY_MEDIUM);
        setGraphicSettingsFPS(FPS_30);
    }

    /**
     * Establece la configuración predeterminada de sonido.
     */
    private void initializeDefaultSoundSettings() {
        setSoundSettingsMasterVolume(10);
    }

    /**
     * Establece la configuración predeterminada de idioma para texto
     * y voz.
     */
    private void initializeDefaultLanguageSettings() {
        setLanguageSettingsText(LANGUAGE_ENGLISH);
        setLanguageSettingsVoice(LANGUAGE_ENGLISH);
    }
}
```

Main

La clase principal **Main** ejemplifica el uso del patrón Singleton mediante dos variables, `settings1` y `settings2`, destinadas a obtener instancias de la clase Settings mediante el método estático `getSettings`. Estas variables son esenciales para verificar si ambas apuntan a la misma instancia, lo cual demuestra la unicidad inherente al patrón Singleton.

Posteriormente, se procede a imprimir la configuración actual utilizando el método `toString` de la clase `Settings`. Esta impresión tiene como propósito confirmar la correcta configuración de las variables.

```
package com.main;

import com.game.config.Settings;
```

```java
/**
 * Clase principal que demuestra el patrón de diseño Singleton.
 */
public class Main {

    public static void main(String[] args) {
        // Obtener instancias de Settings utilizando el método estático getSettings()
        Settings settings1 = Settings.getSettings();
        Settings settings2 = Settings.getSettings();

        // Verificar si solo existe una instancia.
        // Las referencias apuntan al mismo objeto en memoria.
        if (settings1 == settings2)
            System.out.println("Solo hay una instancia de Settings.\n");
        else
            System.out.println("Error: Se han creado múltiples instancias de Settings.\n");

        // Verificar que se configuró adecuadamente.
        System.out.println(settings1);
    }
}
```

Finalmente, se presenta la salida en consola generada durante la ejecución de la clase `Main`.

```
Solo hay una instancia de Settings.

Settings {
graphicSettingsGraphicsQuality = Medium
graphicSettingsResolution = Medium
graphicSettingsFPS = 30
soundSettingsMasterVolume = 10
languageSettingsText = English
languageSettingsVoice = English
}
```

En esta salida, se confirma la existencia de una única instancia de la clase Settings, validando así la efectividad del patrón Singleton. Además, se detallan las configuraciones actuales, subrayando la correcta inicialización de las variables relacionadas con la calidad gráfica, resolución, frecuencia de cuadros, volumen de sonido e idioma.