Universidad del Valle de Guatemala

Campus Central

Departamento de Computación

Algoritmos y Estructuras de datos

Tema:

Hoja de Trabajo 4

Estudiantes: Joel Jaquez y Hugo Barillas.

Carrera: Ingeniería en Ciencias de la Computación y Tecnologías de la info.

Número de Carné: 23369 y 23306

Responder pregunta:
Ventajas y desventajas del Singleton

Ventajas:
Control sobre la instancia: Permite controlar la creación de una única instancia de una clase, asegurando que exista solo una única instancia en toda la aplicación.

Acceso global: Proporciona un punto de acceso global a la instancia de la clase, lo que facilita el acceso a sus métodos y datos desde cualquier parte del código.

Eficiencia de recursos: Al garantizar una única instancia, se evita la sobrecarga de memoria y recursos que podrían surgir al crear múltiples instancias de la misma clase.

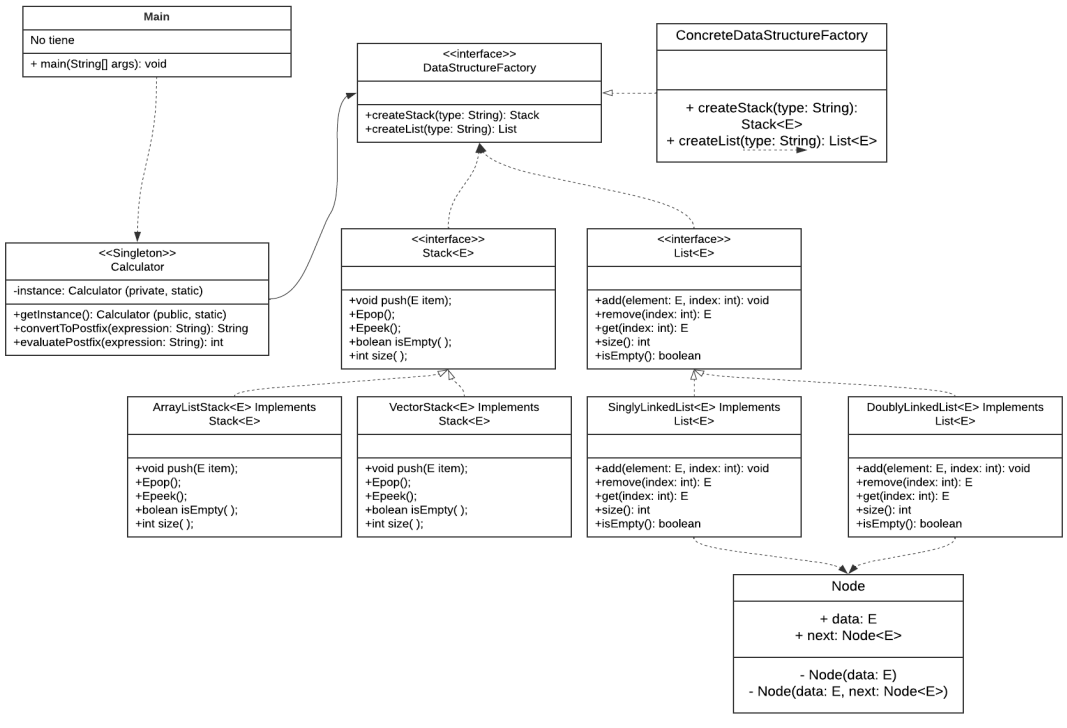
Desventajas:
Acoplamiento fuerte: El uso del patrón Singleton puede introducir un acoplamiento fuerte en el código, lo que dificulta la modularidad y la reutilización de componentes.

Dificultad para pruebas unitarias: Al tener una única instancia global, puede ser difícil realizar pruebas unitarias aisladas, ya que el estado global puede afectar el resultado de las pruebas.

Problemas de concurrencia: En entornos multihilo, se deben tomar precauciones adicionales para garantizar la seguridad en el acceso a la instancia única, lo que puede aumentar la complejidad del código.

En cuanto a su uso en el programa, el Singleton se utiliza para la clase Calculator, asegurando que solo haya una instancia de la calculadora en toda la aplicación. Si bien su comportamiento es similar a una variable global, su uso puede ser adecuado en este contexto si se necesita una única instancia de la calculadora en todo el programa y no se prevén problemas de concurrencia o pruebas unitarias complejas.

Diagrama UML:



Pruebas JUnit:

```java
package uvg.edu.gt;

import static org.junit.Assert.assertTrue;    The import org.junit.Assert.assertTrue is never used

import org.junit.Test;
import static org.junit.Assert.assertEquals;

public class AppTest {

    private Calculator calculator = Calculator.getInstance();

    @Test
    public void testConvertToPostfix() {
        String infix = "(3 + 4) * (5 - 2)";
        String expectedPostfix = "34+52-*";
        assertEquals(expectedPostfix, calculator.convertToPostfix(infix));
    }

    @Test
    public void testEvaluatePostfix() {
        String postfix = "34+52-*";
        int expectedResult = 21;
        assertEquals(expectedResult, calculator.evaluatePostfix(postfix));
    }

    // Aquí puedes agregar más métodos de prueba...
}

```

Enlace al Github:
https://github.com/jaq23369/HojaDeTrabajo4