## Análisis Singleton:

La ventaja del diseño creacional singleton es que nos permite tener un único punto de acceso a un objeto. Es decir, el objeto que implementa el singleton se puede acceder desde cualquier lado pero no puede ser modificado desde cualquier lado; únicamente dentro de su propia clase. Entonces, tenemos una única instancia del objeto pero podemos tener varias referencias al mismo.

Una de las principales desventajas de dicho patrón de diseño es que se desvía del principio de responsabilidad única; puesto que la clase singleton tiene la responsabilidad de crear una instancia de sí misma así como sus otras responsabilidades. Sin embargo, esto se puede solucionar la delegar la parte la creación de instancias a un objeto factory.

Por otro lado, otra desventaja es que las clases singleton no pueden tener sublacses ya que esto violaría el concepto de poder crear una única instancia de la clase singleton. Por último, las clases singleton pueden crear dependencias ocultas en nuestros programas debido a su disponibilidad en cualquier clase.

## Funcionamiento del Programa:

A continuación se muestran capturas de pantalla del funcionamiento de la calculadora Postfix.

```
## src --zsh -- 80x24

42/
34x5+
3*
12x3^
Ingrese una opcion:
8. Salir
1. 12+3*
2. 34-
3. 40/
4. 42/
5. 34x5+
6. 3*
7. 12x3^
8.
Ingrese la opcion de stack que desea implementar:
1. StackUsingArrayList
2. StackUsingVector
3. StackUsingVector
3. StackUsingNeredict
4. StackUsingDoubleLinkedList
5. StackUsingDoubleLinkedList
6. StackUsingDoubleLinkedList
7. StackUsingDoubleLinkedList
8. StackUsingDoubleLinkedList
9. El resultado de la operacion es: 9
1 leonelcontreras@Leonels-MacBook-Pro src %
```

Figura 1: Captura de pantalla Calculadora Postfix

## Pruebas Unitarias:

A continuación, se observan capturas de pantallas de las pruebas unitarias empleadas.

```
J StackFactoryTest.java 2 ×
                                         J StackFactoryTest.java
       Filter (e.g. text, !exclude, @tag)
                                              import static org.junit.Assert.assertEquals;
Q
      10/10 tests passed (100%)
                                              import java.beans.Transient;
       ∨ ⊘ III src 17ms
         import org.junit.Test;
         6
7
8
           public class StackFactoryTest {
           StackFactory factory = new StackFactory<Integer>();
留
           10
11
           O Testdividir() 0.0ms
                                                 @Test
Д

✓ 12
13
14
15
16
✓ 17
18
19
20
21
22
23
24
25
26
✓ 27
28
29
30

                                                  public void TestStackUsingArrayList() {
         > ⊘ ધ LecturaTest 0.0 ▷ 🏚 🖺
                                                     assertEquals(factory.InstanceCreator(
                                                                                                 k: 1) getClass(), StackUsingArrayList.class);
        > @ 😭 StackFactoryTest 17ms
Gr
                                                  public void TestStackUsingVector() {
                                                                                                 k: 2).getClass(), StackUsingVector.class);
                                                     assertEquals(factory.InstanceCreator(ope
                                                 @Test
                                                  public void TestStackUsingLinkedList() {
                                                     assertEquals(factory.InstanceCreator(
                                                                                                   3).getClass(), StackUsingLinkedList.class);
                                                 @Test
                                                  public void TestStackUsindDoubleLinkedList() {
                                                     assertEquals(factory.InstanceCreator(
                                                                                                   4).getClass(), StackUsingDoubleLinkedList.class);
```

Figura 2: Captura de pantalla Test de la clase Factory

```
StackFactoryTest.java 2
                                                                                                                                                                                                                                                                                                                                Filter (e.g. text, !exclude, @tag)
                                                                                                                                                                                                                                                    ort static org.junit.Assert.assertEquals;
                                10/10 tests passed (100%)
                                                                                                                                                                                                                                  import java.beans.Transient;

✓ ✓ III src 17ms

√ Ø {} < Default Package > 17ms

CalculadoraTest 0.0ms

Output

Description:

Output

Descript
                                                        10
                                                                                                                                                                                                                                                    @Test
                                                          O O Testdividir() 0.0ms
                                                                                                                                                                                                                                                      public void TestLectura(){
                                                       12
13
                                                                                                                                                                                                                                                                       assertEquals(" 12*3^", Lectura.In2Post(exp
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 n: "(1*2)^3"<mark>));</mark>
                                          > ⊘ 😭 LecturaTest 0.0 ▷ 🖒 🖰
                                          > 🕢 😭 StackFactoryTest 17ms
  Gr
```

Figura 3: Captura de pantalla Test de la clase Lectura

```
va > 😭 CalculadoraTe
lora miCalculadora
       Filter (e.g. text, !exclude, @tag)
       10/10 tests passed (100%)
                                            10
11
12
13
       ∨ ⊘ III src 17ms
                                                       ublic void Testsumar() {

V (2) { } < Default Package > 17ms

                                                          assertEquals(miCalculadora.Calcular(p
                                                                                                                         : 1),3);
           14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
            O.0ms
            public void Testrestar() {
            assertEquals(miCalculadora.Calcular(
            O (D Testdividir() 0.0ms
            > ⊘ 😭 LecturaTest 0.0 ▷ 🏚 🖰
                                                         blic void Testmultiplicar() {
         > 🕢 ધ StackFactoryTest 17ms
                                                                                                                          (: 1), 35);
                                                          assertEquals(miCalculadora.Calcular(p
Gr
                                                        blic void Testdividir() {
                                                          assertEquals(miCalculadora.Calcular(
                                                                                                                           1), 4);
                                                       public void Testexponente() 🛚
                                                                                                                          : 1), 8);
```

Figura 4: Captura de pantalla Test de la clase Calculadora

## Referencias:

■ Convert Infix expression to Postfix expression (2023) geeksforgeeks.org. Available at: https://www.geeksforgeeks.org/conversion-to-postfix-expression/ (Accessed: February 20, 2023).

■ Emmett, J. (2011) Singleton Pattern-Positive and Negative Aspects, Codeproject. Available at: https://www.codeproject.

- Pattern-Positive-and-Negative-Aspects-2 (Accessed: February 20, 2023).

  Singh N (2018) Let's examine the pros and cons of the Singleton design pattern, freeCodeCamp, Availa-
- Singh, N. (2018) Let's examine the pros and cons of the Singleton design pattern, freeCodeCamp. Available at: https://www.freecodecamp.org/news/singleton-design-pattern-pros-and-cons-e10f98e23d63/ (Accessed: February 20, 2023).