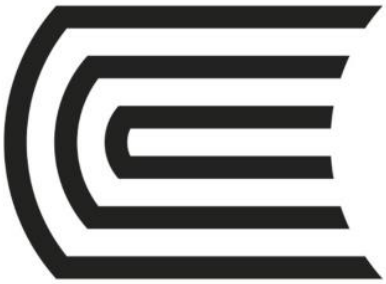
Universidad Continental

“Año del Bicentenario, de la consolidación de nuestra independencia, y de la conmemoración de las heroicas batallas de Junín y Ayacucho”



Universidad Continental

“Técnica de refactorización INLINE METHOD”

Huaccho Mancilla Steven José

Lazo Maravi Nilton Joel

Enlace GitHub: https://github.com/SteveMancilla/TecnicaRefactorizacion_InlineMethod.git

Huancayo - 2024

Técnica de refactorización INLINE METHOD

Está técnica de refactorización consiste en reemplazar llamadas a un método con el contenido del propio método, eliminando así la necesidad de que el método exista por separado.

BENEFICIOS

- **Código más directo:** Eliminar métodos innecesarios hace que el código sea más directo y fácil de entender.
- **Menos saltos:** Reducir la cantidad de saltos que el lector del código debe hacer para entender el flujo del programa.
- **Optimización:** Posible mejora en el rendimiento al eliminar la sobrecarga de llamadas de métodos.

MOTIVACIÓN PARA USAR INLINE METHOD

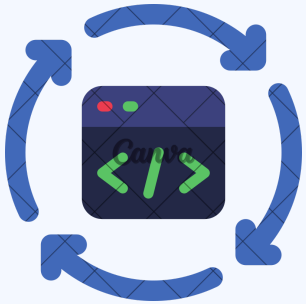
- **Simplificación del código:** Reduce la complejidad innecesaria.
- **Mejora la legibilidad:** Facilita la lectura del código.
- **Reducción de la sobrecarga:** Minimiza el overhead de llamadas de métodos innecesarios.

CUÁNDO APLICARLO

- **Métodos muy simples:** Cuando el método contiene una sola línea de código o realiza una operación muy sencilla.
- **Llamadas Únicas**: Cuando un método se invoca en un solo lugar del código.
- **Métodos Obsoletos**: Cuando un método ya no es relevante en su contexto actual y puede ser reemplazado directamente por su contenido.

CONSIDERACIONES Y PRECAUCIONES

- **Código duplicado:** Asegurarse de que la inlinación no duplique el código en múltiples lugares, lo cual podría hacer el mantenimiento más difícil.
- **Complejidad del código:** Evitar inlinar métodos complejos que podrían hacer el código menos legible.
- **Pruebas unitarias:** Ejecutar pruebas unitarias para garantizar que el comportamiento del código sigue siendo el esperado después de la refactorización.



CONCLUSIÓN

La técnica de refactorización "Inline Method" es útil para simplificar y mejorar la legibilidad del código eliminando métodos innecesarios. Al aplicarla, es esencial considerar el contexto y asegurarse de que no se introduzca duplicación o complejidad innecesaria.

CÓDIGO ANTES DE LA REFACTORIZACIÓN:

```
class Numeros():
    def ingresar_numeros(self):
        numeros = []
        while True:
            try:
                numero = float(input("Ingrese un número (o
cualquier otra tecla para finalizar): "))
                numeros.append(numero)
            except ValueError:
                break
        return numeros

    def separar_numeros(self, numeros):
        pares = []
        impares = []
        for num in numeros:
            if num % 2 == 0:
                pares.append(num)
            else:
                impares.append(num)
        return pares, impares

    def obtener_mayor(self, lista):
        if lista:
            return max(lista)
        else:
            return None

usarclase = Numeros()
#numeros = ingresar_numeros()
numeros = usarclase.ingresar_numeros()
#pares, impares = separar_numeros(numeros)
pares, impares = usarclase.separar_numeros(numeros)

if pares:
    #mayor_par = obtener_mayor(pares)
    mayor_par = usarclase.obtener_mayor(pares)
```

```
    print(f"El mayor número par ingresado es: {mayor_par}")
else:
    print("No se ingresaron números pares.")

if impares:
    #mayor_impar = obtener_mayor(impares)
    mayor_impar = usarclase.obtener_mayor(impares)
    print(f"El mayor número impar ingresado es: {mayor_impar}")
else:
    print("No se ingresaron números impares.")

#mayor_total = obtener_mayor(numeros)
mayor_total = usarclase.obtener_mayor(numeros)
if mayor_total is not None:
    print(f"El mayor número ingresado en total es: {mayor_total}")
else:
    print("No se ingresaron números.")
```

CÓDIGO DESPUÉS DE LA REFACTORIZACIÓN POR LA TÉCNICA: INLINE METHOD

```
numeros = []
pares = []
impares = []

while True:
    try:
        numero = float(input("Ingrese un número (o cualquier otra
tecla para finalizar): "))
        numeros.append(numero)
        pares.append(numero) if numero % 2 == 0 else
impares.append(numero)
    except ValueError:
        break

if pares:
    print(f"El mayor número par ingresado es: {max(pares)}")
else:
    print("No se ingresaron números pares.")

if impares:
    print(f"El mayor número impar ingresado es: {max(impares)}")
else:
    print("No se ingresaron números impares.")

if numeros:
    print(f"El mayor número ingresado en total es:
{max(numeros)}")
else:
    print("No se ingresaron números.")
```

PRUEBAS UNITARIAS:

```python
import unittest
from inlineMethod_ANTES import Numeros

class NumerosParesImpares(unittest.TestCase):

    def setUp(self):
        self.dividir = Numeros()

    def tearDown(self):
        self.dividir = None

    def test_NumeroParImpar(self):

        numeros=[1,2,3,4,5,6,7]
        resultado_esperado_pares = [2,4,6]
        resultado_esperado_impares = [1,3,5,7]

        resultado_actual_pares, resultado_actual_impares =
self.dividir.separar_numeros(numeros)

        self.assertEqual(resultado_esperado_pares,
resultado_actual_pares)
        self.assertEqual(resultado_esperado_impares,
resultado_actual_impares)

if __name__=='__main__':
    unittest.main()
```