Centro Universitário de Brasília – CEUB
Diretoria de Educação a Distância – DIREAD
Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas

Matéria : Verificação e Validação de Software Tuma - C - 0624

Aluno: Felipe Lima da Hora – RA: 72200124

RELATÓRIO DA SISTEMATIZAÇÃO

1 - DA LINGUAGEM DE PROGRAMAÇÃO USADA

Para a sistematização foi usado a linguagem python, na sua versão 3.12 sendo necessário a conversão da classe original em java a linguagem escolhida.

Classe original em Java:

```
public class CalculadoraDiferente {
    public int inverteNumero(int numero){
        int numeroInvertido = 0;
        int temp = 0;

        while(numero > 0){
            temp = numero%10;
            numeroInvertido = numeroInvertido * 10 + temp;
            numero = numero/10;
        }
        return (numeroInvertido);
    }

    public int fatorial(int numero){
        int fatorial = numero;
         for(int i =(numero - 1); i > 1; i--)
         {
            fatorial = fatorial * i;
         }
        return fatorial;
    }

   public int somaDobro(int a, int b) {
        return a + b * 2 ;
    }
```

Classe em python:

```
class CalculadoraDiferente:
    def inverte_numero(self, numero):
        numero_invertido = 0
        while numero > 0:
            temp = numero % 10
            numero_invertido = numero_invertido * 10 + temp
            numero = numero // 10
        return numero_invertido
```

```
    def fatorial(self, numero):
        resultado = 1
        for index in range(1, numero + 1):
            resultado *= index
        return resultado

    def soma_dobro(self, a, b):
        return a + b * 2
```

2 - DA CLASSE DE TESTES

Para fazer os testes foi usada a biblioteca unittest, biblioteca do python análoga ao JUNIT do java, contendo 9 métodos de testes que estão divididos em trios. Cada trio representa um conjunto de testes para os métodos da classe a ser testada, no caso a CalculadoraDiferente.py.

Código:

```
import unittest
from src.CalculadoraDiferente import CalculadoraDiferente

class TestCalculadoraDiferente(unittest.TestCase):

    def setUp(self):
        self.calc = CalculadoraDiferente()

    #CASO 1 (INVERTER NUMEROS)

    def test_inverte_numero_1(self):
        self.assertEqual(self.calc.inverte_numero(123), 321)

    def test_inverte_numero_2(self):
        self.assertEqual(self.calc.inverte_numero(1111), 1111)

    def test_inverte_numero_3(self):
        self.assertEqual(self.calc.inverte_numero(120), 21)
```

```
    # CASO 2 (FATORIAL)
    def test_fatorial_1(self):
        self.assertEqual(self.calc.fatorial(5), 120)

    def test_fatorial_2(self):
        self.assertEqual(self.calc.fatorial(0), 1)

    def test_fatorial_3(self):
        self.assertEqual(self.calc.fatorial(1), 1)

    # CASO 3 (SOMA O DOBRO DO SEGUNDO NÚMERO)
    def test_soma_dobro_1(self):
        self.assertEqual(self.calc.soma_dobro(2, 3), 8)

    def test_soma_dobro_2(self):
        self.assertEqual(self.calc.soma_dobro(4, 0), 4)

    def test_soma_dobro_3(self):
        self.assertEqual(self.calc.soma_dobro(-1, -1), -3)

def start_test():
    if __name__ == 'main':
        unittest.main()
```

3 - DOS TESTES

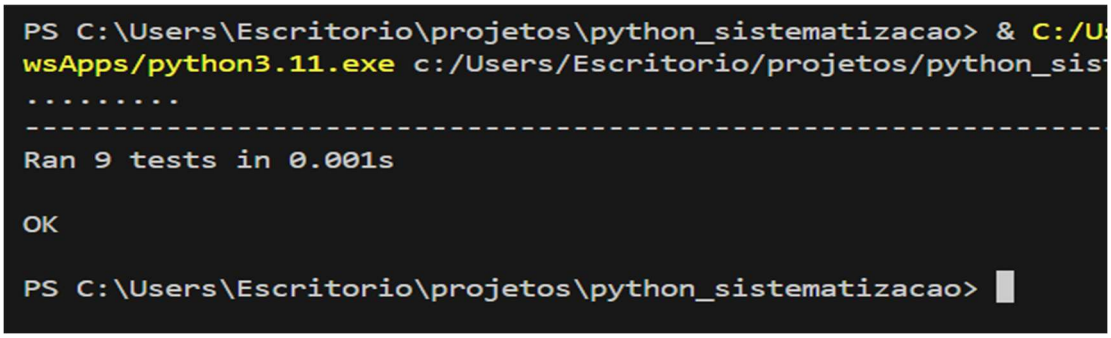
No primeiro trio de métodos da classe de testes, separado pelo comentário #CASO 1 (INVERTER NUMEROS), foram escolhidas as entradas 123, 1111 e 120, o primeiro se dá em virtude de ser um caso corriqueiro de número, o segundo pelo fato de possuir 4 números e ser idêntico e por fim o último por conter um 0.

Como previsão de saída temos para 123 a saída 321, para 1111 a saída 1111 e para 120 a saída 21.

No segundo trio, separado pelo comentário #CASO 2 (FATORIAL), foram escolhidos os números 5, que saída deve ser 120, e os número 0 e 1 onde a saída de ambos deve ser 1.

Por fim no último trio de testes, apontado abaixo do comentário # CASO 3 (SOMA O DOBRO DO SEGUNDO NÚMERO), os números escolhidos foram 2 | 3 que a saída deve ser 8, 4 | 0 que a saída deve 4 e -1|-1 que a saída deve ser -3.

Ao rodar os testes todos passaram com sucesso, demonstrando que o código inicial atendeu aos requisitos do teste.



4 - DA EXECUÇÃO

Existem duas formas de executar a aplicação, contudo é necessário que o sistema operacional possua o python devidamente instalado.

4.1 – Pelo Terminal:

Basta acessar a pasta com código pelo terminal e usar o comando “python3 test_main.py”

4.2 – Execução direta:

Basta dar dois cliques no arquivo test_main.py que terminal irá ser abrir por 3 segundos.

5 - DA CONCLUSÃO

Com os métodos de testes é possível verificar se a saída do código é a desejada de acordo com a entrada, este teste é muito bom para verificar se determinada classe ou método continua integra depois de modificações.