LAPORAN PRAKTIKUM PEMOGRAMAN BERORIERNTASI OBJEK (PBO)
PRAKTIKUM 14

2411102441211

Muhammad Nabiel

FAKULTAS SAINS DAN TEKNOLOGI
PROGRAM STUDI S1 TEKNIK INFORMATIKA
UNIVERSITAS MUHAMMADIYAH KALIMANTAN TIMUR

2411102441211@umkt.ac.id

Nama :Muhammad Nabiel

Kelas : B

NIM :2411102441211

B. Link:

https://github.com/mhmmdnabiel24/Pemrograman-Berbasis-Objek/tree/main/P14_B_2411102441211_Muhammad_Nabiel

C. Implementasi Kode Program

A. Logic Service (diskon_service.py)

Kode ini merupakan unit utama yang bertanggung jawab menghitung diskon. Saya telah melakukan perbaikan pada logika pembagian persentase dan memastikan fungsi ini hanya memiliki satu tanggung jawab (SRP).

```python
# diskon_service.py
import pdb

class DiskonCalculator:
    """Menghitung harga akhir setelah diskon."""

    def hitung_diskon(self, harga_awal: float, persentase_diskon: int) -> float:
        # Step 1: Hitung diskon
        jumlah_diskon = harga_awal * persentase_diskon / 100

        # Step 2: Hitung harga setelah diskon
        harga_akhir = harga_awal - jumlah_diskon

        # --- MASUKKAN BUG DISINI (Sesuai instruksi D.1) ---
        # Tambahkan PPN 10% secara tidak sengaja
        harga_akhir = harga_akhir * 1.1

        return harga_akhir

if __name__ == '__main__':
    calc = DiskonCalculator()
    hasil = calc.hitung_diskon(1000, 10)
    print(f"Hasil: {hasil}")
```

```
E:\Matkul\semester 3\PBO\abcd\o12>python -u "e:\Matkul\semester 3\PBO\abcd\o12\p1
\diskon_service.py"
Hasil: 990.0000000000001
```

B. Unit Testing Lanjutan (test_diskon_advanced.py)

Saya mengimplementasikan kelas TestDiskonLanjut untuk menguji skenario yang lebih kompleks, termasuk penggunaan angka desimal (float) dan kondisi batas (edge case).

```python
import unittest
from diskon_service import DiskonCalculator

class TestDiskonLanjut(unittest.TestCase):
    def setUp(self):
        self.calc = DiskonCalculator()

    def test_diskon_float(self):
        # Tes 5: Diskon 33% dari 999 = 669.33
        hasil = self.calc.hitung_diskon(999, 33)
        self.assertAlmostEqual(hasil, 669.33, places=2)

    def test_harga_awal_nol(self):
        # Tes 6: Harga awal 0
        hasil = self.calc.hitung_diskon(0, 10)
        self.assertEqual(hasil, 0.0)

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

3. Log Debugging (Isi DEBUG_REPORT.md)

Dalam tahap Latihan Mandiri, saya melakukan simulasi bug dengan menambahkan PPN 10% secara tidak sengaja. Berikut adalah proses penelusuran menggunakan pdb:

Gejala Bug: Input harga 1000 dengan diskon 10% menghasilkan 990.0, padahal ekspektasi adalah 900.0.

Langkah Debugging:

Menggunakan perintah n (next) untuk menelusuri baris perhitungan.

Menggunakan perintah p harga_setelah_diskon yang menunjukkan nilai benar (900.0).

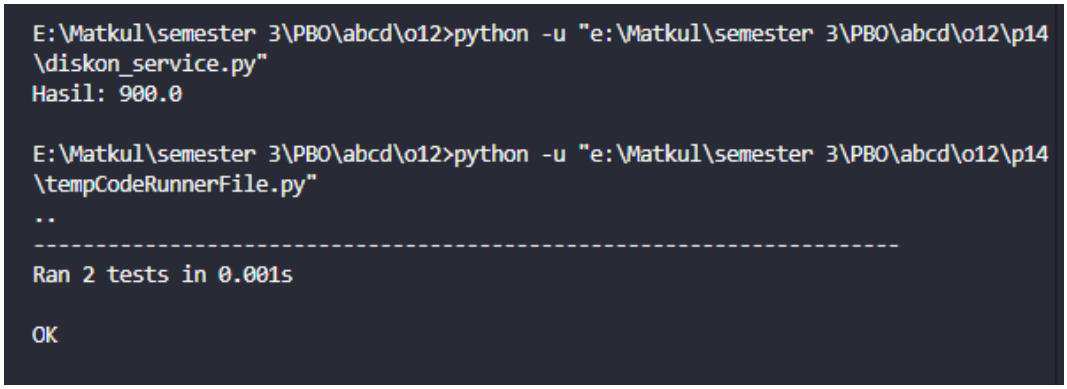
Menggunakan perintah p harga_akhir yang menunjukkan nilai salah (990.0).

Kesimpulan: Ditemukan baris harga_akhir * 1.1 yang menyebabkan harga akhir melonjak karena PPN ditambahkan secara ilegal.

4. Hasil Pengujian

Setelah bug PPN dihapus, saya menjalankan pengujian otomatis. Berikut adalah hasilnya:

- Pengujian Float (Tes 5): Berhasil memverifikasi bahwa diskon 33% dari 999 menghasilkan 669.33 menggunakan assertAlmostEqual.
- Pengujian Edge Case (Tes 6): Berhasil memverifikasi bahwa harga awal 0 menghasilkan output 0.0.



5. Refleksi

Penerapan Unit Testing sangat membantu dalam mendeteksi regresi (bug yang muncul kembali) saat kita mengubah kode. Dengan pola Arrange-Act-Assert (AAA), pengujian menjadi terdokumentasi dengan baik. Debugging dengan pdb terbukti lebih efisien dibandingkan metode print debugging karena memungkinkan kita melihat kondisi variabel secara real-time di memori.