LAPORAN PRAKTIKUM JURNAL MODUL 12

NIM / Nama: 2311104072 – Jauhar Fajar Zuhair

Bagian I: Pendahuluan

Laporan Jurnal Modul 12 ini mendokumentasikan proses pembuatan aplikasi Windows Forms sederhana untuk menghitung hasil perpangkatan dan pengujian unit (Unit Testing) terhadap logika perhitungannya menggunakan framework MSTest. Tujuan utama adalah memastikan fungsi kalkulasi pangkat (`CalculatePowerValue`) bekerja dengan benar sesuai dengan berbagai aturan dan kasus uji yang ditentukan, termasuk kasus-kasus tepi (edge cases).

Bagian II: Aplikasi Windows Forms (`MainForm.cs`)

Aplikasi ini menyediakan antarmuka pengguna grafis (GUI) untuk memasukkan bilangan basis (`a`) dan eksponen (`b`), serta menampilkan hasil perhitungan pangkat.

Antarmuka Pengguna (UI) - Berdasarkan `MainForm.Designer.cs`:

- `textBoxBase`: Kotak teks untuk memasukkan nilai basis (`a`).
- `textBoxExponent`: Kotak teks untuk memasukkan nilai eksponen (`b`).
- `buttonCalculate`: Tombol untuk memicu perhitungan.
- `labelResult`: Label untuk menampilkan hasil perhitungan atau pesan error.
- Label-label lain (`labelBase`, `labelExponent`, `labelTitle`) untuk informasi tambahan.

Logika Aplikasi (`MainForm.cs`):

1. Event Handler Tombol Hitung (`buttonCalculate_Click`)

```
private void buttonCalculate_Click(object sender, EventArgs e)
{
    // Mencoba membaca input dari TextBox
    if (double.TryParse(textBoxBase.Text, out double baseValue) &&
        double.TryParse(textBoxExponent.Text, out double exponentValue))
    {
        try
        {
            // Memanggil fungsi kalkulasi utama
            double result = CalculatePowerValue(baseValue, exponentValue);
            // Menampilkan hasil jika berhasil
            labelResult.Text = $"Hasil: {result}";
        }
        catch (ArgumentException ex)
        {
```

```
            // Menampilkan pesan error spesifik dari CalculatePowerValue
            labelResult.Text = $"Error: {ex.Message}";
        }
        catch (Exception ex)
        {
            // Menampilkan pesan error umum lainnya
            labelResult.Text = $"Error: Terjadi kesalahan tidak terduga
({ex.Message})";
        }
    }
    else
    {
        // Menampilkan pesan jika input tidak valid
        labelResult.Text = "Error: Input basis dan eksponen harus berupa
angka.";
    }
}
```

Penjelasan:

- Metode ini dieksekusi ketika `buttonCalculate` diklik.
- Ia mencoba mengonversi teks dari `textBoxBase` dan `textBoxExponent` menjadi tipe `double`.
- Jika konversi berhasil, ia memanggil metode `CalculatePowerValue`.
- Hasil perhitungan atau pesan error (jika terjadi `Exception`) ditampilkan di `labelResult`.
- Jika input awal tidak valid (bukan angka), pesan error ditampilkan.

2. Fungsi Kalkulasi Inti (`CalculatePowerValue`)

```
internal double CalculatePowerValue(double a, double b)
{
    // Aturan ii: Jika a = 0 dan b > 0, hasil 0.
    if (a == 0 && b > 0)
    {
        return 0;
    }
    // Aturan ii: Jika a = 0 dan b <= 0, tidak terdefinisi (throw exception).
    else if (a == 0 && b <= 0)
    {
        throw new ArgumentException("Basis tidak boleh nol jika eksponen nol
atau negatif.");
    }
    // Aturan i: Jika b = 0, hasil selalu 1 (bahkan jika a = 0, ditangani
Math.Pow).
    // Aturan iii & iv: Menggunakan Math.Pow untuk kasus lainnya (termasuk b
< 0).
    return Math.Pow(a, b);
}
```

Penjelasan:

- Metode ini menerima basis `a` dan eksponen `b` sebagai input.
- Metode ini (berdasarkan implementasinya dan aturan yang diuji) menerapkan beberapa aturan:
    - Jika basis `a` adalah 0 dan eksponen `b` positif, hasilnya 0.
    - Jika basis `a` adalah 0 dan eksponen `b` nol atau negatif, ini dianggap tidak terdefinisi dan melempar `ArgumentException`.
- Metode ini ditandai `internal` agar bisa diakses oleh proyek test (karena `MainForm.cs` kemungkinan tidak memiliki `InternalsVisibleTo`). *Catatan: Jika `MainForm.cs` memiliki `InternalsVisibleTo`, metode ini bisa `private` atau `internal`.*

Bagian III: Unit Tests (`PowerCalculatorTests.cs`)

File ini berisi serangkaian test case menggunakan MSTest (`Microsoft.VisualStudio.TestTools.UnitTesting`) untuk memvalidasi metode `CalculatePowerValue`.

Struktur Test Class:

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using jmmodul12_2311104072; // Namespace dari aplikasi utama

[TestClass] // Menandakan kelas ini berisi test MSTest
public class PowerCalculatorTests
{
    private MainForm _calculator; // Instance MainForm untuk pengujian

    [TestInitialize] // Dijalankan sebelum setiap test method
    public void Setup()
    {
        _calculator = new MainForm(); // Membuat instance baru
    }

    [TestCleanup] // Dijalankan setelah setiap test method
    public void Cleanup()
    {
        _calculator?.Dispose(); // Membersihkan resource form
    }

    // ... Test Methods ...
}
```

Penjelasan:

- `[TestClass]`: Atribut yang menandakan kelas ini untuk MSTest.
- `[TestInitialize]`: Metode `Setup` membuat instance baru `MainForm` sebelum setiap test, memastikan isolasi antar test.

- `[TestCleanup]`: Metode `Cleanup` memanggil `Dispose` pada `MainForm` setelah setiap test untuk membersihkan resource.

Test Cases (Berdasarkan Aturan):

1. Aturan i: Jika b = 0, selalu return 1

```
[TestMethod] // Menandakan ini adalah test method
public void CalculatePowerValue_ExponentZero_ReturnsOne()
{
    // Arrange, Act, Assert dalam satu baris untuk kesederhanaan
    Assert.AreEqual(1, _calculator.CalculatePowerValue(5, 0), "5^0 harus 1");
    Assert.AreEqual(1, _calculator.CalculatePowerValue(-3, 0), "(-3)^0 harus
1");
    // Kasus a=0, b=0 ditangani oleh Math.Pow yang mengembalikan 1
    Assert.AreEqual(1, _calculator.CalculatePowerValue(0, 0), "0^0 harus 1
(sesuai Math.Pow)");
}
```

Penjelasan: Menguji berbagai basis dengan eksponen 0, memastikan hasilnya selalu 1.

2. Aturan ii: Jika a = 0

```
[TestMethod]
public void CalculatePowerValue_BaseZeroExponentPositive_ReturnsZero()
{
    Assert.AreEqual(0, _calculator.CalculatePowerValue(0, 5), "0^5 harus 0");
}

[TestMethod]
public void
CalculatePowerValue_BaseZeroExponentZeroOrNegative_ThrowsArgumentException()
{
    // Memastikan exception dilempar untuk 0^0 (meskipun CalculatePowerValue
mengandalkan Math.Pow untuk ini,
    // tapi implementasi internalnya melempar exception jika b <= 0)
    Assert.ThrowsException<ArgumentException>(() =>
_calculator.CalculatePowerValue(0, 0));
    // Memastikan exception dilempar untuk 0^-2
    Assert.ThrowsException<ArgumentException>(() =>
_calculator.CalculatePowerValue(0, -2));
}
```

Penjelasan: Menguji kasus basis 0. Jika eksponen positif, hasil 0. Jika eksponen nol atau negatif, diharapkan `ArgumentException` dilempar (sesuai logika `CalculatePowerValue`).

3. Aturan iii: Jika b < 0, return 1 / (a ^ |b|)

```
[TestMethod]
public void CalculatePowerValue_NegativeExponent_ReturnsReciprocal()
{
```

```
    // Arrange
    double baseVal = 2;
    double exponentVal = -3;
    double expected = 1.0 / (Math.Pow(baseVal, Math.Abs(exponentVal))); // 1
/ (2^3) = 1/8 = 0.125

    // Act
    double actual = _calculator.CalculatePowerValue(baseVal, exponentVal);

    // Assert (menggunakan delta untuk perbandingan double)
    Assert.AreEqual(expected, actual, 1e-9, "2^-3 harus 0.125");

    // Contoh lain
    Assert.AreEqual(1.0/9.0, _calculator.CalculatePowerValue(3, -2), 1e-9,
"3^-2 harus 1/9");
}
```

Penjelasan: Menguji eksponen negatif. Hasilnya harus sama dengan 1 dibagi basis pangkat nilai absolut eksponen. Perbandingan `double` menggunakan `Assert.AreEqual` dengan toleransi (delta).

4. Aturan iv: Kasus Umum (a != 0, b != 0, b > 0)

```
[TestMethod]
public void CalculatePowerValue_ValidInputs_ReturnsCorrectPower()
{
    Assert.AreEqual(8, _calculator.CalculatePowerValue(2, 3), "2^3 harus 8");
    Assert.AreEqual(9, _calculator.CalculatePowerValue(-3, 2), "(-3)^2 harus
9");
    Assert.AreEqual(-27, _calculator.CalculatePowerValue(-3, 3), "(-3)^3
harus -27");
    Assert.AreEqual(100, _calculator.CalculatePowerValue(10, 2), "10^2 harus
100");
}
```

Penjelasan: Menguji berbagai kombinasi input basis dan eksponen positif/negatif yang valid, memastikan hasil sesuai dengan perhitungan pangkat standar.

Bagian IV: Menjalankan Test dan Hasil