Tentu, berikut adalah dokumentasi lengkap dari kode LogicalOperations.cs, yang menjelaskan fungsionalitas dan alasan desainnya baris per baris (atau lebih tepatnya, blok per blok).

Dokumentasi: LogicalOperations.cs

Ringkasan File

- **Namespace:** MiniPhotoshop.Logic.ImageProcessing
- **Kelas:** LogicalOperations
- **Tujuan:** Kelas ini adalah *utility class* (kelas pembantu) statis yang menyediakan metode untuk melakukan operasi logika Boolean (AND, OR, XOR, NOT) pada gambar.
- **Fitur Utama:** Operasi ini bersifat **biner**. Artinya, sebelum membandingkan dua piksel, setiap piksel pertama-tama dikonversi menjadi hitam murni (nilai 0) atau putih murni (nilai 255) berdasarkan ambang batas kecerahan (grayscale).
- **Catatan Performa:** Sama seperti ArithmeticOperations.cs dan ConstantOperations.cs, kelas ini menggunakan metode GetPixel dan SetPixel yang **lambat**, tidak seperti metode LockBits yang lebih cepat yang terlihat di Brightness.cs atau BlackWhite.cs.

Analisis Kode per Blok

Blok 1: Definisi Kelas dan using

C#

```
using System;
using System.Drawing;
```

```
namespace MiniPhotoshop.Logic.ImageProcessing
{
    // ... (XML summary comments) ...
    public static class LogicalOperations
    {
```

- **using System; / using System.Drawing;**: Mengimpor *namespace* dasar. System diperlukan untuk Func<> dan System.Drawing diperlukan untuk Bitmap dan Color.
- **public static class LogicalOperations**: Mendeklarasikan kelas *utility* yang bersifat public dan static.
    - **Alasan:** Ini adalah *helper class* standar. static berarti Anda tidak perlu membuat instansi (new LogicalOperations()) dan dapat memanggil metodenya secara langsung, seperti LogicalOperations.And(imgA, imgB).

Blok 2: Metode Pembantu Internal

C#

```
        // 1. Metode Pembantu Internal

        private static int ToBinary(Color c)
        {
            int gray = (int)(c.R * 0.299 + c.G * 0.587 + c.B * 0.114);
            return (gray < 128) ? 0 : 255;
        }

        private static Color ToBinaryColor(int val)
        {
            return Color.FromArgb(val, val, val);
        }
```

- **private static int ToBinary(Color c)**
    - **Penjelasan:** Ini adalah fungsi inti yang mengubah piksel berwarna menjadi nilai biner (hitam/putih).
    - **Fungsi:**
        1. int gray = (int)(c.R * ...): Mengonversi warna RGB menjadi satu nilai *grayscale* (skala keabuan) menggunakan formula luminans standar (bobot berbeda untuk

R, G, dan B).
    2. return (gray < 128) ? 0 : 255;: Ini adalah **binarisasi threshold**. Jika nilai keabuan di bawah 128 (lebih gelap dari abu-abu tengah), anggap sebagai hitam (0). Jika tidak (128 atau lebih terang), anggap sebagai putih (255).
  - **Alasan:** Operasi logika (seperti AND, OR) memerlukan nilai "benar" (true) dan "salah" (false) yang jelas. Di sini, 255 (putih) mewakili "benar" dan 0 (hitam) mewakili "salah".
- **private static Color ToBinaryColor(int val)**
  - **Penjelasan:** Fungsi pembantu kebalikan dari ToBinary.
  - **Fungsi:** Mengambil nilai biner (0 atau 255) dan mengubahnya kembali menjadi Color yang terlihat.
  - **Alasan:** Color.FromArgb(0, 0, 0) adalah Color.Black. Color.FromArgb(255, 255, 255) adalah Color.White. Ini diperlukan untuk menyimpan hasil biner kembali ke Bitmap.

Blok 3: Mesin Operasi Utama PerformOperation

C#

```
    // 2. Metode Privat Utama

    private static Bitmap PerformOperation(Bitmap imgA, Bitmap imgB, Func<int, int, int> operation)
    {
        int newWidth = Math.Max(imgA.Width, imgB.Width);
        // ... (baris lainnya) ...
        Bitmap resultBmp = new Bitmap(newWidth, newHeight);

        for (int y = 0; y < newHeight; y++)
        {
            for (int x = 0; x < newWidth; x++)
            {
                // ... (logika bool inA, inB) ...

                if (inA && inB)
                {
                    // KASUS 1: Piksel saling bertemu
                    int valA = ToBinary(imgA.GetPixel(x, y));
                    int valB = ToBinary(imgB.GetPixel(x, y));
                    resultBmp.SetPixel(x, y, ToBinaryColor(operation(valA, valB)));
```

```
            }
            else if (inA)
            {
                // KASUS 2: Piksel hanya ada di gambar A
                resultBmp.SetPixel(x, y, imgA.GetPixel(x, y));
            }
            // ... (else if inB dan else) ...
        }
    }
    return resultBmp;
}
```

- **private static Bitmap PerformOperation(...)**
  - **Penjelasan:** Ini adalah metode inti yang mengabstraksi logika perulangan (looping) piksel, mirip dengan yang ada di ArithmeticOperations.cs.
  - **Fungsi:** Menerima dua gambar (A dan B) dan sebuah "fungsi operasi" (Func<int, int, int> operation). Fungsi ini (operation) akan memberi tahu metode ini *apa* yang harus dilakukan (AND, OR, XOR) pada dua nilai piksel biner.
  - **Alasan (Prinsip DRY):** Desain ini **(Don't Repeat Yourself)** sangat baik. Daripada menulis *looping* for (x...)/for (y...) dan logika if (inA && inB) yang rumit sebanyak tiga kali (untuk AND, OR, XOR), logika itu ditulis *satu kali* di sini.
- **Logika if (inA && inB)**
  - **Penjelasan:** Ini adalah inti dari logika. Hanya ketika piksel ada di *kedua* gambar, operasi logika dilakukan.
  - **Fungsi:**
    1. int valA = ToBinary(imgA.GetPixel(x, y));: Mengambil piksel A, mengubahnya menjadi biner (0 atau 255).
    2. int valB = ToBinary(imgB.GetPixel(x, y));: Mengambil piksel B, mengubahnya menjadi biner (0 atau 255).
    3. operation(valA, valB): Menjalankan fungsi yang dioper (misal, fungsi AND) pada dua nilai biner tersebut.
    4. ToBinaryColor(...): Mengubah hasil biner (misal, 255) kembali menjadi Color (misal, putih).
    5. resultBmp.SetPixel(...): Menyimpan Color hasil ke gambar baru.
- **Logika else if (inA) / else if (inB)**
  - **Penjelasan:** Jika piksel hanya ada di salah satu gambar (karena gambar satunya lebih kecil), operasi logika tidak dilakukan.
  - **Fungsi:** Piksel dari gambar yang lebih besar tersebut disalin begitu saja ke hasil akhir.

Blok 4: Metode Publik (API)

C#

```
    // 3. Metode Publik (API)

    public static Bitmap And(Bitmap imgA, Bitmap imgB)
    {
        return PerformOperation(imgA, imgB, (valA, valB) =>
        {
            return (valA == 255 && valB == 255) ? 255 : 0;
        });
    }

    public static Bitmap Or(Bitmap imgA, Bitmap imgB)
    {
        // ... (implementasi OR) ...
    }

    public static Bitmap Xor(Bitmap imgA, Bitmap imgB)
    {
        // ... (implementasi XOR) ...
    }
```

- **public static Bitmap And(Bitmap imgA, Bitmap imgB)**
  - **Penjelasan:** Metode publik untuk operasi AND.
  - **Fungsi:** Memanggil PerformOperation dan memberinya *lambda expression* ((valA, valB) => ...) sebagai parameter operation.
  - **Alasan (Lambda):** Lambda ini mendefinisikan *apa* arti "AND" dalam konteks biner ini: "Hanya kembalikan 255 (putih/true) jika valA adalah 255 **DAN** valB adalah 255. Jika tidak, kembalikan 0 (hitam/false)."
- **public static Bitmap Or(...)**: Sama, tetapi logikanya (valA == 255 || valB == 255) (kembalikan putih jika valA **ATAU** valB putih).
- **public static Bitmap Xor(...)**: Sama, tetapi logikanya (valA != valB) (kembalikan putih jika valA dan valB **BERBEDA**—satu putih dan satu hitam).

Blok 5: Metode Publik Not

C#

```
    public static Bitmap Not(Bitmap imgA)
    {
        Bitmap resultBmp = new Bitmap(imgA.Width, imgA.Height);

        for (int y = 0; y < imgA.Height; y++)
        {
            for (int x = 0; x < imgA.Width; x++)
            {
                int valA = ToBinary(imgA.GetPixel(x, y));
                int valRes = (valA == 255) ? 0 : 255; // Balikkan nilai biner
                resultBmp.SetPixel(x, y, ToBinaryColor(valRes));
            }
        }

        return resultBmp;
    }
```

- **public static Bitmap Not(Bitmap imgA)**
  - **Penjelasan:** Operasi NOT (negasi biner) pada *satu* gambar.
  - **Alasan (Tidak Pakai PerformOperation):** Metode ini tidak menggunakan PerformOperation karena NOT adalah operasi *unary* (hanya butuh satu input), sedangkan PerformOperation dirancang untuk operasi *binary* (dua input gambar).
  - **Fungsi:**
    1. Looping manual melalui setiap piksel imgA.
    2. int valA = ToBinary(...): Mengubah piksel menjadi 0 atau 255.
    3. int valRes = (valA == 255) ? 0 : 255;: Ini adalah **inti logika NOT**. Nilai biner dibalik: 255 (putih) menjadi 0 (hitam), dan 0 (hitam) menjadi 255 (putih).
    4. resultBmp.SetPixel(...): Menyimpan warna yang sudah dibalik.