LAPORAN PRAKTIKUM

ALGORITMA DAN PEMOGRAMAN

OPERATOR ARITMATIKA GUI

PEKAN 8

Disusun Oleh :

KINAYA NOVRYA MANDA

(2511531016)

Dosen Pengampu :

DR. WAHYUDI, S.T, M.T

Asisten Praktikum :

MUHAMMAD ZAKI AL HAFIZ

DEPARTEMEN INFORMATIKA

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

2025

KATA PENGANTAR

Puji syukur penulis sampaikan kehadirat Allah SWT. Salawat dan salam disampaikan kepada Nabi Muhammad SAW. Karena thaufik dan hidayah-Nya, laporan praktikum Java ini dapat diselesaikan dengan baik dan tepat waktu. Laporan ini disusun sebagai salah satu bentuk pertanggungjawaban dari kegiatan praktikum yang telah dilaksanakan, sekaligus sebagai sarana untuk memperdalam pemahaman mengenai konsep dasar serta penerapan bahasa pemrograman Java.

Melalui praktikum ini, penulis memperoleh pengalaman langsung dalam memahami hubungan antara logika perhitungan dan desain antarmuka yang saling mendukung menggunakan Java. Diharapkan laporan ini dapat memberikan gambaran yang jelas mengenai materi yang telah dipelajari serta hasil dari percobaan yang dilakukan selama praktikum berlangsung.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna, baik dari segi isi maupun penyajiannya. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi perbaikan di masa yang akan datang.

Akhir kata, penulis mengucapkan terima kasih kepada dosen pengampu, asisten laboratorium, serta semua pihak yang telah memberikan bimbingan, arahan, dan dukungan sehingga laporan praktikum ini dapat tersusun. Semoga laporan ini dapat bermanfaat bagi pembaca, khususnya dalam memahami dasar-dasar pemrograman Java.

Padang, November 2025

Penulis

..

DAFTAR ISI

...

DAFTAR LAMPIRAN

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Pemrograman merupakan salah satu dasar penting dalam dunia teknologi, dan di dalamnya terdapat konsep operator aritmatika yang berfungsi untuk melakukan berbagai jenis perhitungan. Operator seperti penjumlahan, pengurangan, perkalian, dan pembagian menjadi fondasi dalam membangun logika program. Namun, pemahaman tentang operator aritmatika tidak cukup hanya dipelajari secara teoritis, sehingga diperlukan praktikum untuk melihat bagaimana operator tersebut bekerja secara langsung dalam kode.

Selain memahami operator aritmatika, mahasiswa juga perlu mengenal *Graphical User Interface* (GUI) sebagai cara membuat aplikasi yang lebih mudah digunakan oleh pengguna. GUI memberikan tampilan visual berupa tombol, kotak input, dan hasil keluaran sehingga proses perhitungan tidak lagi dilakukan melalui terminal teks, melainkan melalui tampilan yang lebih interaktif. Penggabungan operator aritmatika dengan GUI membantu mahasiswa melihat bagaimana logika program dapat diterapkan ke dalam aplikasi yang memiliki interaksi nyata dengan pengguna.

Melalui praktikum ini, mahasiswa dapat memahami hubungan antara logika perhitungan dan desain antarmuka yang saling mendukung. Praktikum ini juga memudahkan mahasiswa untuk mengembangkan keterampilan dalam membuat program sederhana yang mampu menerima input, memprosesnya dengan operator aritmatika, lalu menampilkan hasilnya secara visual. Dengan demikian, praktikum ini menjadi langkah awal untuk membangun kemampuan pemrograman yang lebih kompleks di masa mendatang.

1.2 Tujuan Praktikum

Tujuan dilakukannya praktikum ini adalah sebagai berikut :

1. Memahami cara kerja operator aritmatika dalam pemrograman.
2. Mempelajari implementasi operator aritmatika ke dalam aplikasi berbasis GUI.
3. Melatih penggunaan komponen GUI seperti *input box,* tombol, dan *output display.*
4. Menghubungkan logika perhitungan dengan *event* pada GUI (misalnya *event* klik tombol).
5. Menumbuhkan kemampuan dalam membuat aplikasi sederhana yang interaktif dan mudah digunakan.

1.3 Manfaat Praktikum

Manfaat dilakukannya praktikum ini adalah sebagai berikut :

1. Mahasiswa dapat memahami konsep operator aritmatika secara lebih aplikatif.
2. Mampu membuat program sederhana yang dapat melakukan perhitungan dengan tampilan GUI.
3. Mahasiswa mengenal dasar-dasar pembuatan antarmuka visual pada aplikasi.
4. Melatih logika pemrograman dan pemahaman alur *input*–proses–*output*.
5. Menjadi dasar untuk mempelajari topik lanjutan validasi input dan pembuatan aplikasi yang lebih kompleks.

BAB II

PEMBAHASAN

2.1 Kode Program

```
1  package Pekan8_2511531016;
2  import java.awt.BorderLayout;
19
20 public class OperatorAritmatikaGUI_2511531016 extends JFrame {
21
22     private static final long serialVersionUID = 1L;
23     private JPanel contentPane;
24     private JTextField txtBil2;
25     private JTextField txtHasil;
26     private JTextField txtBil1;
27
28
29     private void pesanPeringatan(String pesan) {
30         JOptionPane.showMessageDialog(this, pesan,"Peringatan", JOptionPane.WARNING_MESSAGE);
31     }
32     private void pesanEror(String pesan) {
33         JOptionPane.showMessageDialog(this, pesan,"Kesalahan", JOptionPane.ERROR_MESSAGE);
34     }
35
36     /**
37      * Launch the application.
38      */
39     public static void main(String[] args) {
40         EventQueue.invokeLater(new Runnable() {
41             public void run() {
42                 try {
43                     OperatorAritmatikaGUI_2511531016 frame = new OperatorAritmatikaGUI_2511531016();
44                     frame.setVisible(true);
45                 } catch (Exception e) {
46                     e.printStackTrace();
47                 }
48             }
49         });
50     }
51
52     /**
53      * Create the frame.
54      */
55     public OperatorAritmatikaGUI_2511531016() {
56         setTitle("OPERATOR ARITMATIKA");
57         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
58         setBounds(100, 100, 450, 300);
59         contentPane = new JPanel();
60         contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
61         setContentPane(contentPane);
62         contentPane.setLayout(null);
63
64         JLabel lblNewLabel_1 = new JLabel("Bilangan 1");
65         lblNewLabel_1.setBounds(10, 45, 51, 14);
66         contentPane.add(lblNewLabel_1);
67
68         JLabel lblNewLabel = new JLabel("OPERATOR ARITMATIKA");
69         lblNewLabel.setBounds(165, 10, 161, 18);
70         lblNewLabel.setFont(new Font("Tempus Sans ITC", Font.BOLD, 13));
71         contentPane.add(lblNewLabel);
72
73         JLabel lblNewLabel_2 = new JLabel("Bilangan 2");
74         lblNewLabel_2.setBounds(10, 70, 51, 14);
```

```
75          contentPane.add(lblNewLabel_2);
76
77          JLabel lblNewLabel_3 = new JLabel("Operator");
78          lblNewLabel_3.setBounds(-17, 108, 72, 18);
79          lblNewLabel_3.setHorizontalAlignment(SwingConstants.TRAILING);
80          contentPane.add(lblNewLabel_3);
81
82          JLabel lblNewLabel_4 = new JLabel("Hasil");
83          lblNewLabel_4.setBounds(13, 169, 48, 14);
84          contentPane.add(lblNewLabel_4);
85
86          txtBil2 = new JTextField();
87          txtBil2.setHorizontalAlignment(SwingConstants.CENTER);
88          txtBil2.setBounds(114, 67, 72, 20);
89          contentPane.add(txtBil2);
90          txtBil2.setColumns(10);
91
92          txtHasil = new JTextField();
93          txtHasil.setHorizontalAlignment(SwingConstants.CENTER);
94          txtHasil.setBounds(114, 166, 72, 20);
95          contentPane.add(txtHasil);
96          txtHasil.setColumns(10);
97
98          txtBil1 = new JTextField();
99          txtBil1.setHorizontalAlignment(SwingConstants.CENTER);
100         txtBil1.setBounds(114, 42, 72, 20);
101         contentPane.add(txtBil1);
102         txtBil1.setColumns(10);
```

```
103
104         JComboBox cbOperator = new JComboBox();
105         cbOperator.setModel(new DefaultComboBoxModel(new String[] {"+", "-", "*", "/", "%"}));
106         cbOperator.setBounds(114, 106, 72, 22);
107         contentPane.add(cbOperator);
108
109         JButton btnNewButton = new JButton("Proses");
110         btnNewButton.addActionListener(new ActionListener() {
111             int hasil;
112             public void actionPerformed(ActionEvent e) {
113                 if(txtBil1.getText().trim().isEmpty()) {
114                     pesanPeringatan("Bilangan 1 harus diisi");
115                 } else if (txtBil2.getText().trim().isEmpty()) {
116                     pesanPeringatan("Bilangan 1 harus diisi");
117                 } else {
118                     try {
119                         int a = Integer.parseInt(txtBil1.getText());
120                         int b = Integer.parseInt(txtBil2.getText());
121                         int c = cbOperator.getSelectedIndex(); //operator
122                         if (c==0) {
123                             hasil = a+b;
124                         }
125                         if (c==1) {
126                             hasil = a-b;
127                         }
128                         if (c==2) {
129                             hasil = a*b;
130                         }
131                         if (c==3) {
```

```
132                             hasil = a/b;
133                         }
134                         if (c==4) {
135                             hasil = a%b;
136                         }
137                     } catch (NumberFormatException ex) {
138                         pesanEror("Bilangan 1 dan Bilangan 2 harus angka");
139                     }
140                 }
141
142                 txtHasil.setText(String.valueOf(hasil));
143             }
144         });
145         btnNewButton.setBounds(207, 106, 89, 23);
146         contentPane.add(btnNewButton);
147
148     }
149 }
```

Kode Program 2.1 (Kode Program OperatorAritmatikaGUI)

2.1.1 Uraian Kode Program :

1. *private void pesanPeringatan(String pesan) { JOptionPane.showMessageDialog(this, pesan, "Peringatan", JOptionPane.WARNING_MESSAGE); }*

→ Method ini digunakan untuk menampilkan kotak dialog peringatan. Setiap kali *method* dipanggil, program akan memunculkan pesan dengan icon *warning* dan judul “Peringatan”.

2. *private void pesanEror(String pesan) { JOptionPane.showMessageDialog(this, pesan, "Kesalahan", JOptionPane.ERROR_MESSAGE); }*

→ Method ini menampilkan kotak dialog *error*. Dialog yang muncul bertipe *ERROR_MESSAGE*, biasanya untuk memberi tahu bahwa input tidak valid.

3. Variabel *int hasil;*

→ Variabel ini digunakan untuk menyimpan hasil operasi aritmatika yang diproses nantinya.

4. public void actionPerformed(ActionEvent e) {

→ Method ini berjalan ketika tombol pada GUI ditekan. Semua proses perhitungan akan dilakukan di dalam *method* ini.

5. if (txtBil1.getText().trim().isEmpty()) {
pesanPeringatan("Bilangan 1 harus diisi"); }

→ Mengecek apakah input bilangan pertama kosong. Jika kosong, tampil pesan peringatan.

6. int a = Integer.parseInt(txtBil1.getText());int b = Integer.parseInt(txtBil2.getText());

→ Mengkonversi teks dari input menjadi tipe data *integer* agar bisa dihitung.

7. int c = cbOperator.getSelectedIndex();

→ Mengambil indeks operator yang dipilih pada *ComboBox,* misalnya 0 = tambah, 1 = kurang, 2 = kali, dst.

8. } catch (NumberFormatException ex) {
pesanEror("Bilangan 1 dan Bilangan 2 harus angka"); }

→ Jika input bukan angka (misalnya huruf atau simbol), program langsung memunculkan pesan *error*.

9. txtHasil.setText(String.valueOf(hasil));

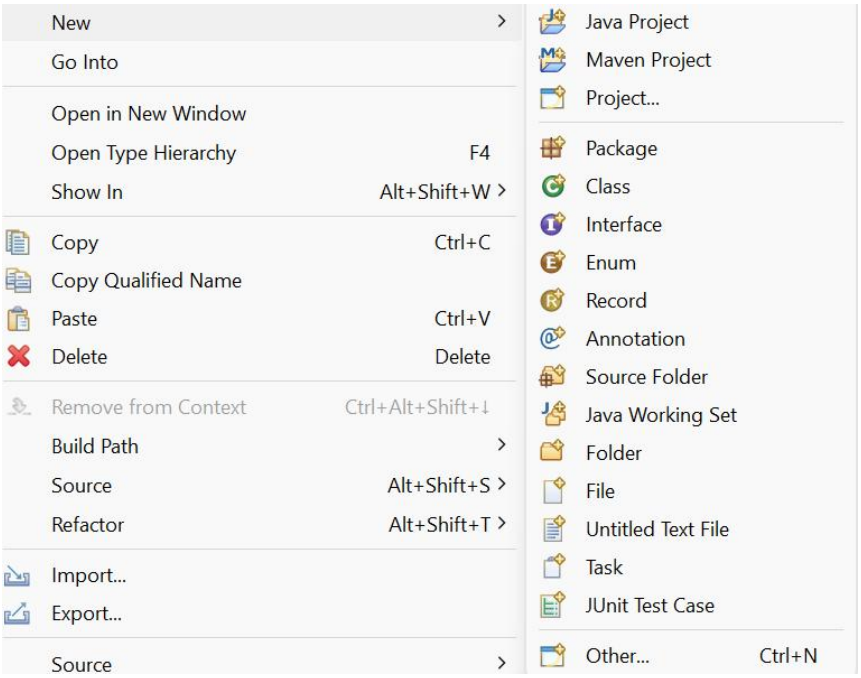
→ Setelah perhitungan selesai, hasil operasi ditampilkan pada *field* hasil dalam bentuk teks.

2.2 Langkah Kerja

1. Membuat *package* bernama *Pekan8_2511531016.*

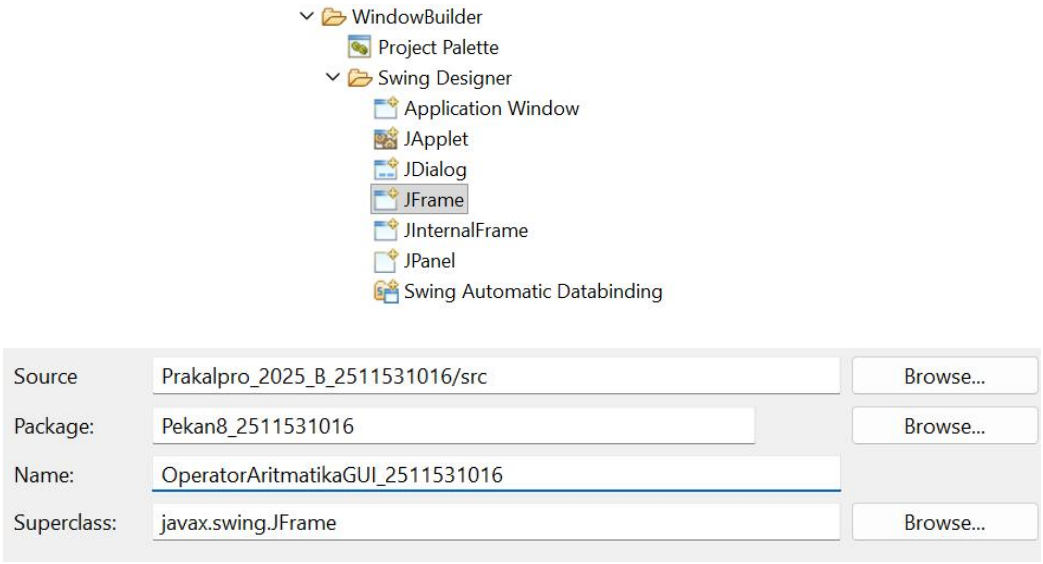
Gambar 2.1 (Langkah 1 OperatorAritmatikaGUI)

2. Klik kanan pada package dan pilih *new* lalu klik *other.*



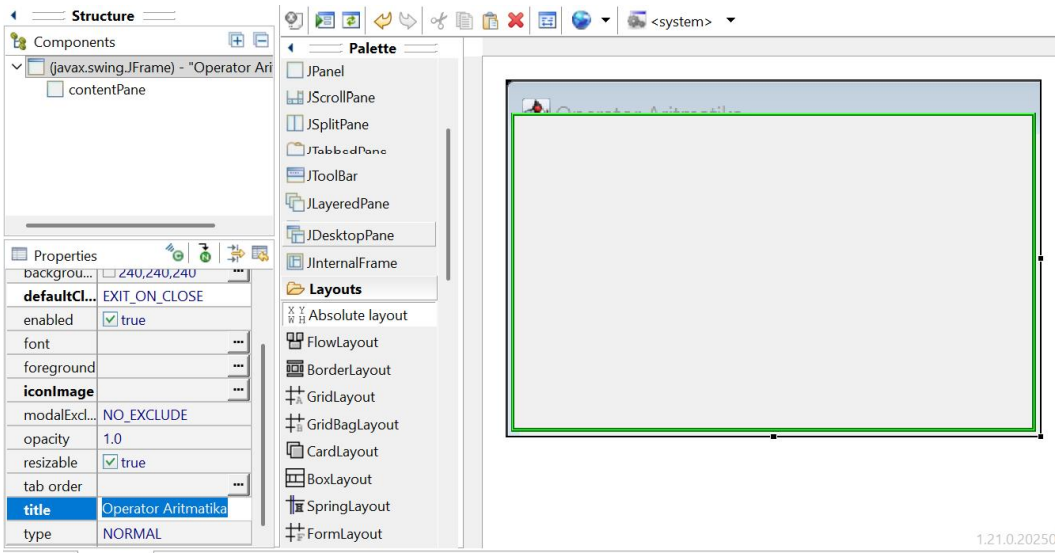
Gambar 2.2 (Langkah 2 OperatorAritmatikaGUI)

3. Klik *JFrame* pada menu *WindowBuilder* dan beri nama *OperatorAritmatikaGUI_2511531016.*



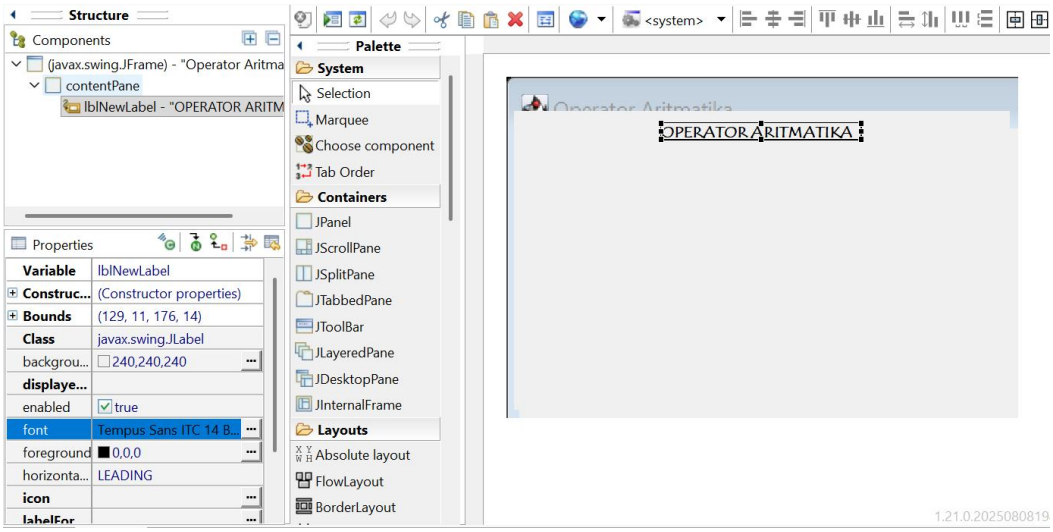
Gambar 2.3 (Langkah 3 OperatorAritmatikaGUI)

4. Akan muncul tampilan seperti ini, lalu buat judul (*title*) yaitu Operator Aritmatika. Setelah itu, klik *contenPane* lalu ubah *layout* nya menjadi *absolute layout.*



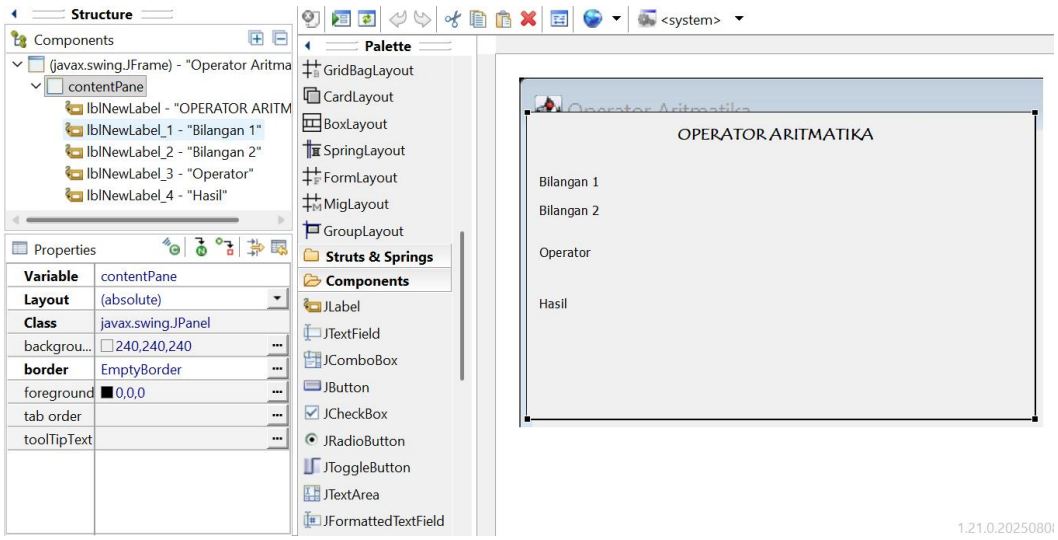
Gambar 2.4 (Langkah 4 OperatorAritmatikaGUI)

5. Pada menu *component* klik *JLabel* lalu buat *NewLabel* berjudul “OPERATOR ARITMATIKA”. Pilih gaya *font* dan ukuran teks sesuai keinginan.



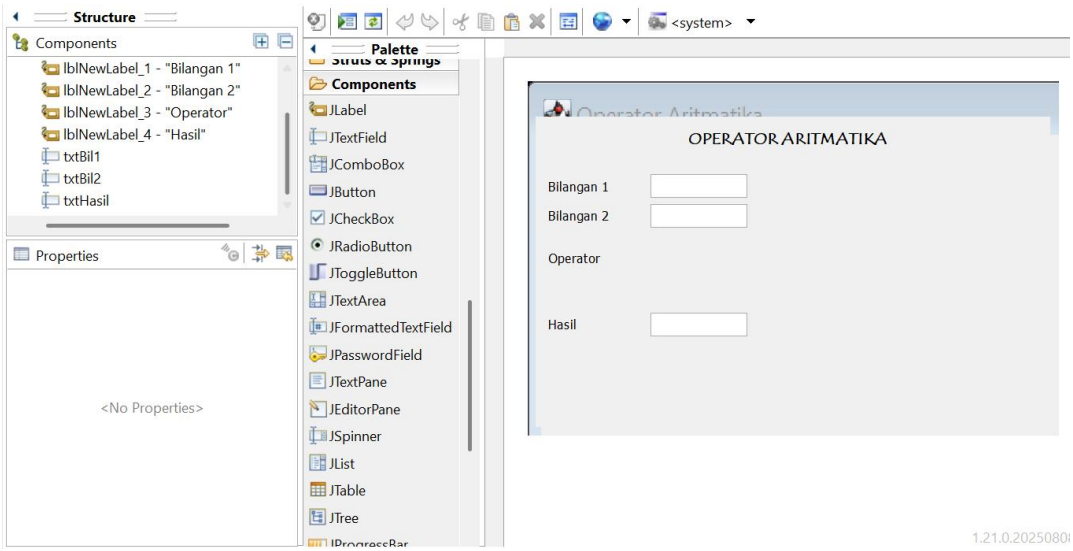
Gambar 2.5 (Langkah 5 OperatorAritmatikaGUI)

6. Tambahkan lagi *JLabel* untuk membuat teks kedalam *design* aplikasi yaitu “Bilangan 1”, “Bilangan 2”, “Operator”, dan “Hasil”.



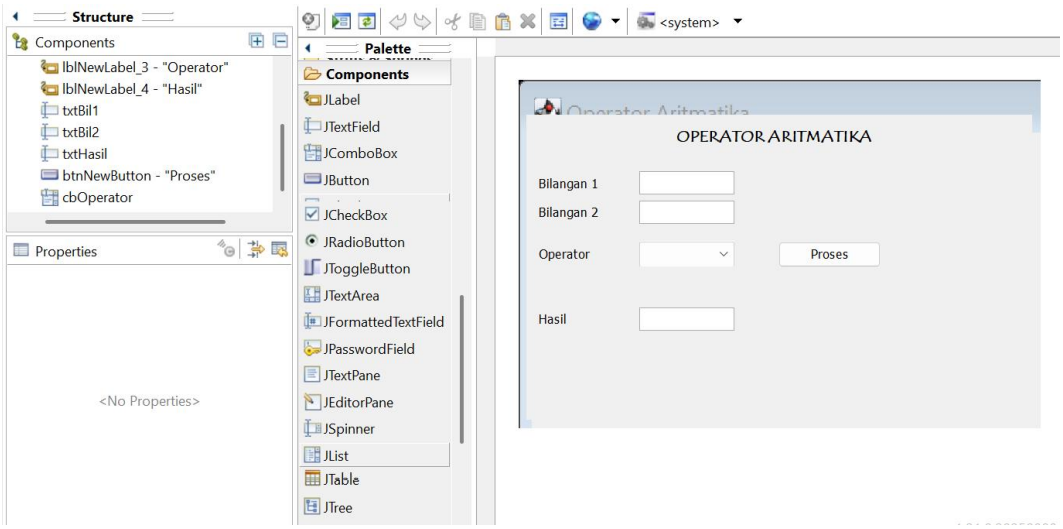
Gambar 2.6 (Langkah 6 OperatorAritmatikaGUI)

7. Tambahkan *JTextField* disamping kanan label Bilangan 1, Bilangan 2 dan Hasil, lalu ubah variabel nya menjadi “txtBil1”, “txtBil2”, dan “txtHasil”.



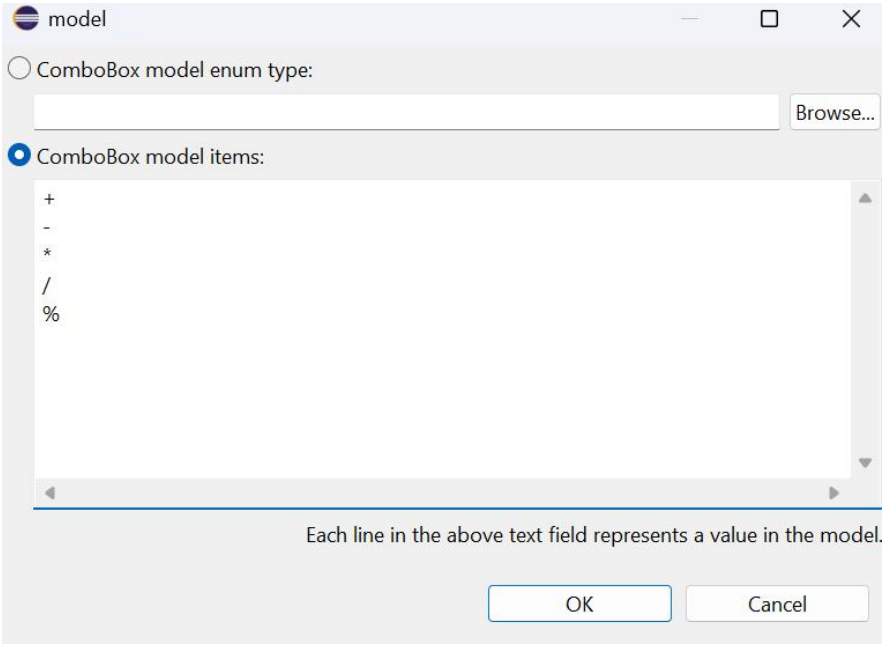
Gambar 2.7 (Langkah 7 OperatorAritmatikaGUI)

8. Tambahkan *JComboBox* disamping kanan label Operator, lalu ubah variabel nya menjadi "cbOperator”. Setelah itu, tambahkan *JButton* lalu ubah teks menjadi “Proses”.



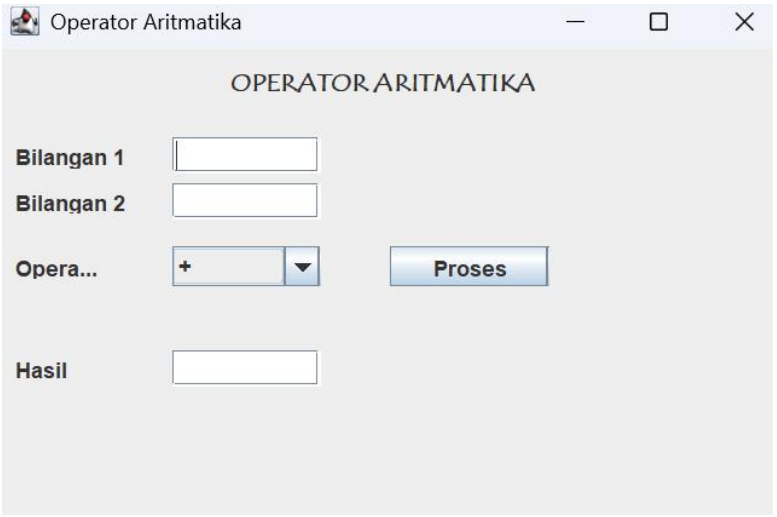
Gambar 2.8 (Langkah 8 OperatorAritmatikaGUI)

9. Tambahkan operator-operator aritmatik (+ - * / %) pada model.



Gambar 2.9 (Langkah 9 OperatorAritmatikaGUI)

10. Tampilan akan menjadi seperti ini, lalu kita akan lanjutkan ke *source code* untuk menjalankan program.



Gambar 2.10 (Langkah 10 OperatorAritmatikaGUI)

11. Tambahkan kode program *pesanPeringatan* dan *pesanKesalahan* seperti dibawah ini jika ada kesalahan input.

```
private void pesanPeringatan(String pesan) {
    JOptionPane.showMessageDialog(this, pesan,"Peringatan", JOptionPane.WARNING_MESSAGE);
}
private void pesanEror(String pesan) {
    JOptionPane.showMessageDialog(this, pesan,"Kesalahan", JOptionPane.ERROR_MESSAGE);
}
```

Gambar 2.11 (Langkah 11 OperatorAritmatikaGUI)

12. Untuk menjalankan program, buat kode dibawah ini, lalu program sudah bisa dijalankan.

```
int hasil;
public void actionPerformed(ActionEvent e) {
    if(txtBil1.getText().trim().isEmpty()) {
        pesanPeringatan("Bilangan 1 harus diisi");
    } else if (txtBil2.getText().trim().isEmpty()) {
        pesanPeringatan("Bilangan 1 harus diisi");
    } else {
        try {
            int a = Integer.parseInt(txtBil1.getText());
            int b = Integer.parseInt(txtBil2.getText());
            int c = cbOperator.getSelectedIndex(); //operator
            if (c==0) {
                hasil = a+b;
            }
            if (c==1) {
                hasil = a-b;
            }
            if (c==2) {
                hasil = a*b;
            }
            if (c==3) {
                hasil = a/b;
            }
            if (c==4) {
                hasil = a%b;
            }
        } catch (NumberFormatException ex) {
            pesanEror("Bilangan 1 dan Bilangan 2 harus angka");
        }
    }

    txtHasil.setText(String.valueOf(hasil));
```

Gambar 2.12 (Langkah 12 OperatorAritmatikaGUI)

2.3 Analisis Hasil

Berdasarkan hasil percobaan dari kode program yang telah dijalankan, dapat dilihat output yang ditampilkan menunjukkan bahwa program operator aritmatika telah menerapkan validasi input dengan baik. Saat pengguna memasukkan data yang tidak sesuai, seperti karakter "7k" pada kolom bilangan, program langsung menampilkan pesan kesalahan berupa *popup*. Hal ini membuktikan bahwa program mampu mendeteksi input yang tidak valid sebelum melakukan proses perhitungan, sehingga mencegah terjadinya *error* dan menjaga alur program tetap stabil. Validasi ini sangat penting agar aplikasi tidak salah memproses data atau menampilkan hasil yang tidak akurat.

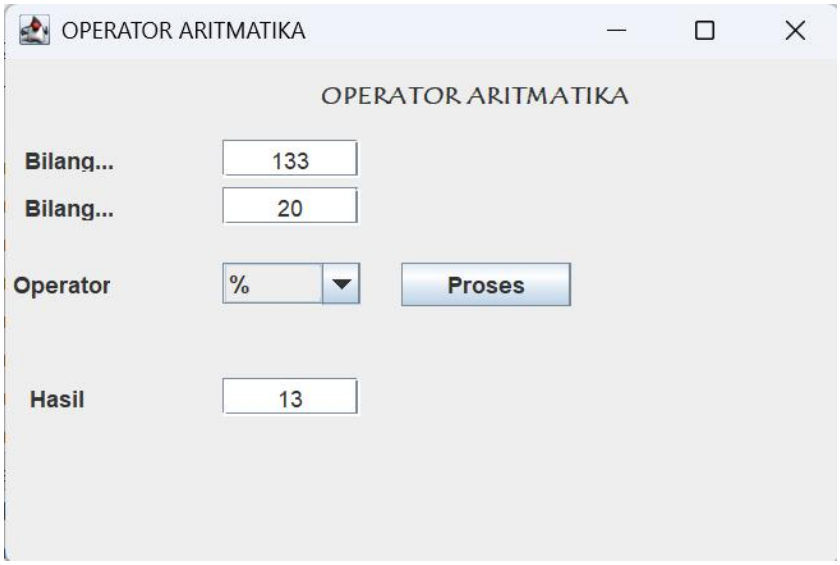
Selain itu, penggunaan *popup* sebagai media pemberitahuan membuat pesan kesalahan menjadi jelas dan mudah dipahami oleh pengguna. Pesan seperti "Bilangan 1 dan Bilangan 2 harus angka" memberi informasi yang spesifik mengenai apa yang harus diperbaiki, sehingga pengguna dapat langsung memperbaiki inputnya tanpa kebingungan. Penyampaian pesan secara langsung ini juga sesuai dengan prinsip *user-friendly*, di mana aplikasi harus memberikan umpan balik yang cepat dan informatif ketika terjadi kesalahan dalam pengisian data.

Berdasarkan hasil tersebut, dapat disimpulkan bahwa program ini sudah memenuhi prinsip dasar pengembangan perangkat lunak, terutama dalam hal validasi input dan penanganan kesalahan. Dalam teori pemrograman, validasi input diperlukan untuk memastikan data sesuai dengan tipe yang diharapkan agar proses perhitungan berjalan benar. Selain itu, pemberian pesan kesalahan termasuk bagian dari prinsip *usability* pada desain antarmuka. Dengan demikian, penerapan validasi dan *feedback* pada program ini sudah sesuai dengan dasar teori dan mendukung keandalan aplikasi secara keseluruhan.

BAB III
KESIMPULAN

3.1 Hasil Praktikum

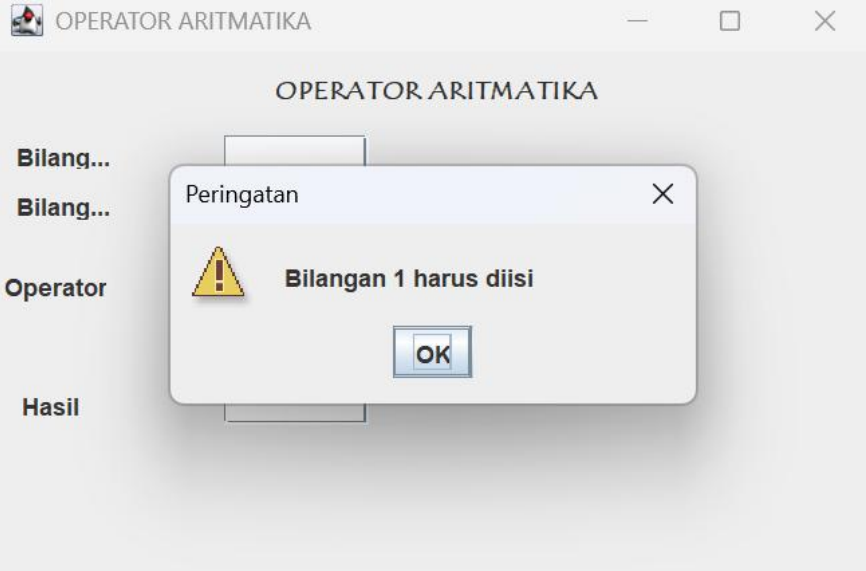
3.1.1 Output Berhasil



Gambar 3.1 (Hasil Praktikum Output Berhasil)

Output di atas menampilkan hasil perhitungan dari program "Operator Aritmatika", di mana pengguna memasukkan dua angka yaitu 133 dan 20, lalu memilih operator % (*modulus*). Operator *modulus* digunakan untuk mencari sisa hasil pembagian antara dua bilangan. Jadi ketika tombol Proses ditekan, program menghitung *133 % 20*, dan menghasilkan nilai 13 sebagai sisanya. Hasil ini kemudian langsung ditampilkan pada kolom "Hasil". Program ini membantu pengguna memahami cara kerja operasi aritmatika sederhana dalam aplikasi Java GUI.

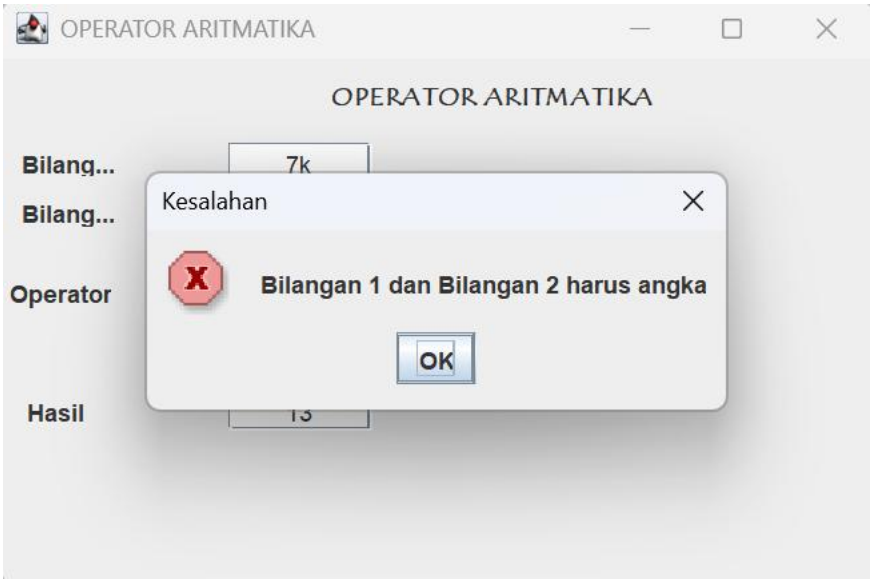
3.1.2 Output Peringatan



Gambar 3.2 (Hasil Praktikum Output Peringatan)

Output di atas menunjukkan bahwa program menampilkan pesan peringatan karena pengguna menekan tombol Proses tanpa mengisi nilai pada bagian Bilangan 1. Sistem otomatis mengecek apakah input sudah lengkap, dan ketika kolom Bilangan 1 kosong, muncul *popup* berisi pesan "Bilangan 1 harus diisi". Peringatan ini membantu pengguna supaya tidak melakukan perhitungan dengan data yang belum lengkap, sehingga program bisa berjalan dengan benar dan tidak menimbulkan *error*.

3.1.3 Output Kesalahan



Gambar 3.3 (Hasil Praktikum Output Kesalahan)

Output di atas menunjukkan bahwa program menampilkan pesan kesalahan karena pengguna memasukkan nilai yang bukan angka pada salah satu kolom input, misalnya menulis “7k” pada Bilangan 1. Program otomatis memeriksa apakah kedua input benar-benar berupa angka sebelum melakukan perhitungan. Jika ada karakter lain selain angka, maka muncul *popup* dengan pesan “Bilangan 1 dan Bilangan 2 harus angka”. Peringatan ini memastikan supaya pengguna memasukkan data yang benar, sehingga proses perhitungan tidak error dan hasil yang ditampilkan tetap valid.

3.2 Saran Pengembangan

Berdasarkan hasil analisis dari kode program yang telah dijalankan, terdapat beberapa saran pengembangan agar program menjadi lebih interaktif, efisien, dan bermanfaat dalam penerapan nyata.

Untuk pengembangan ke depan, program operator aritmatika ini dapat ditambah dengan fitur operasi matematika yang lebih lengkap, seperti pangkat, akar kuadrat, persentase, atau operasi kombinasi. Dengan adanya variasi operasi yang lebih banyak, program akan menjadi lebih fleksibel dan dapat digunakan untuk kebutuhan perhitungan yang lebih kompleks. Selain itu, bagian validasi input juga dapat dibuat lebih detail, misalnya dengan menandai kolom mana yang salah agar pengguna lebih mudah memperbaiki kesalahan.

Pengembangan selanjutnya bisa dilakukan pada tampilan antarmuka (GUI). Desain aplikasi dapat dibuat lebih modern dengan menambahkan warna, ikon, atau tata letak yang lebih rapi agar terlihat profesional dan mudah dipahami oleh pengguna pemula. Selain itu, tombol-tombol dapat diberikan efek visual atau animasi ringan untuk meningkatkan pengalaman pengguna. Dengan tampilan yang lebih informatif dan menarik, aplikasi akan terasa lebih nyaman dan enak digunakan.

Terakhir, program ini dapat dikembangkan menjadi lebih fungsional dengan menambahkan fitur riwayat perhitungan, penyimpanan hasil ke file (seperti .txt atau .csv), dan fitur reset otomatis. Program juga dapat dikembangkan menjadi aplikasi berbasis web atau mobile agar lebih mudah diakses di berbagai perangkat. Dengan pengembangan seperti ini, program tidak hanya berguna untuk tugas praktikum, tetapi juga dapat menjadi aplikasi kecil yang bermanfaat dalam kehidupan sehari-hari.

DAFTAR PUSTAKA

[1] P. J. Deitel and H. M. Deitel, *Java How to Program, Early Objects*, 10th ed. Pearson, 2015. [Online]. https://openlibrary.telkomuniversity.ac.id/pustaka/117192/java-how-to-program-early-objects-global-edition-10-e

[2] H. Schildt, *Java: The Complete Reference*, 9th ed. McGraw-Hill, 2014. [Online]. https://openlibrary.telkomuniversity.ac.id/pustaka/117218/java-the-complete-reference-9-e-

[3] J. Nielsen, *Usability Engineering*. Morgan Kaufmann, 1994. [Online]. https://openlibrary.org/books/OL7328356M/Usability_Engineering_%28Interactive_Technologies%29