

LAPORAN PRAKTIKUM
ALGORITMA DAN PEMROGRAMAN
PENERAPAN *GRAPHICAL USER INTERFACE (GUI)* DALAM
PEMROGRAMAN *JAVA*



Disusun Oleh:

Haliya Isma Husna Putri Ahmadi
2511532002

Dosen Pengampu:
Wahyudi. Dr. S.T.M.T

Asisten Praktikum:
Muhammad Zaky Al Hafiz

DEPARTEMEN INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS ANDALAS

2025

KATA PENGANTAR

Puji dan Syukur penulis panjatkan ke hadirat Tuhan Yang Maha Esa karena atas Rahmat dan karunia-Nya, penulis dapat menyelesaikan lapora praktikum Algoritma dan Pemrograman pekan 8 dengan judul “Penerapan *Graphical User Interface (GUI)* dalam Pemrograman *Java*” tepat pada waktunya.

Penyusunan laporan ini bertujuan untuk memenuhi salah satu tugas dalam mata kuliah Praktikum Algoritma dan Pemrograman, sekaligus sebagai sarana pembelajaran bagi penulis dalam memahami penerapan antarmuka grafis atau *Graphical User Interface (GUI)* pada bahasa pemrograman *Java*. Melalui praktikum ini, penulis dapat mempelajari bagaimana menggunakan komponen - komponen *GUI* seperti *JFrame*, *JButton*, *JLabel*, dan *JTextField*.

Penulis menyadari bahwa laporan ini masih jauh dari sempurna. Oleh karena itu, penulis sangat terbuka terhadap kritik dan saran yang membangun demi perbaikan di masa yang akan datang.

Ucapan terima kasih yang sebesar-besarnya penulis sampaikan kepada dosen pengampu, asisten praktikum, serta semua pihak yang terlibat dalam pelaksanaan praktikum dan penyusunan laporan ini. Semoga laporan ini dapat memberikan manfaat dan menambah wawasan bagi penulis maupun pembaca.

Padang, 17 November 2025

Penulis

DAFTAR ISI

KATA PENGANTAR.....	ii
DAFTAR ISI	iii
BAB I PENDAHULUAN.....	4
1.1 Latar Belakang	4
1.2 Tujuan	4
1.3 Manfaat	4
BAB II PEMBAHASAN	5
2.1 Dasar Teori.....	5
2.1.1 <i>WindowBuilder</i>	5
2.1.2 <i>Java Swing</i>	6
2.1.3 <i>Container J-Frame</i>	6
2.1.4 <i>Component GUI Java Swing</i>	6
2.1.5 <i>Event Handling</i>	7
2.2 Langkah Kerja	8
2.2.1 Pembuatan <i>Package</i> dan <i>Class</i>	8
2.2.2 <i>Upload</i> ke GitHub.....	27
BAB III KESIMPULAN.....	30
3.1 Kesimpulan	30
3.2 Saran	30
DAFTAR PUSTAKA.....	31

BAB I

PENDAHULUAN

1.1 Latar Belakang

Dalam perkembangan teknologi perangkat lunak, kebutuhan akan aplikasi yang interaktif dan mudah digunakan semakin meningkat. Salah satu cara untuk memenuhi kebutuhan tersebut adalah dengan menerapkan *Graphical User Interface (GUI)* pada aplikasi. Berbeda dengan program berbasis teks (*console*), *GUI* memungkinkan pengguna berinteraksi melalui elemen visual seperti tombol, kotak teks, jendela, dan menu, sehingga pengalaman penggunaan menjadi lebih intuitif dan efisien.

Melalui praktikum ini, mahasiswa diharapkan dapat memahami cara merancang dan mengimplementasikan *GUI* menggunakan *Java Swing*, mulai dari pembuatan jendela utama (*JFrame*), penambahan komponen seperti *JButton*, *JLabel*, dan *JTextField*, hingga pemanfaatan event handling untuk mengatur interaksi antara pengguna dan program.

1.2 Tujuan

Tujuan dari pelaksanaan praktikum ini adalah sebagai berikut:

- 1.2.1 Memahami konsep dasar *Graphical User Interface (GUI)* dalam pemrograman *Java*.
- 1.2.2 Mampu menggunakan komponen-komponen *GUI* seperti *JFrame*, *JButton*, *JLabel*, *JTextField*, dan lainnya
- 1.2.3 Menguasai penggunaan *WindowBuilder* dalam merancang tampilan *GUI* secara efisien.

1.3 Manfaat

Manfaat yang diharapkan dari pelaksanaan praktikum ini antara lain:

- 1.3.1 Memberikan pemahaman mengenai pembuatan antarmuka grafis menggunakan *Java Swing*.
- 1.3.2 Memberikan pengalaman langsung dalam merancang dan mengimplementasikan komponen *GUI* pada aplikasi atau program *Java* .

BAB II PEMBAHASAN

2.1 Dasar Teori

Dalam bahasa pemrograman *Java*, *Graphical User Interface (GUI)* merupakan antarmuka yang memungkinkan *user* berinteraksi dengan aplikasi/ program melalui elemen visual seperti tombol, jendela, ikon, dan *text box*. Dalam pemrograman *Java*, *GUI* biasanya dibuat menggunakan *Java Swing*, yaitu *library* yang menyediakan berbagai komponen untuk menyusun tampilan aplikasi. Dengan adanya *GUI*, aplikasi menjadi lebih mudah digunakan dibandingkan aplikasi berbasis teks (*console*), karena pengguna dapat melakukan *input* dan navigasi menggunakan komponen visual.

Pada praktikum ini, mahasiswa mempelajari cara merancang dan mengimplementasikan *GUI* menggunakan *Java Swing* serta memanfaatkan *WindowBuilder*, yaitu *plugin visual* editor untuk membuat tampilan i secara *drag-and-drop* di *Eclipse*. Pemahaman terhadap *JFrame*, komponen *GUI*, serta mekanisme *event handling* menjadi dasar penting dalam membangun aplikasi /program *Java* yang interaktif.

2.1.1 *WindowBuilder*

WindowBuilder adalah sebuah *GUI designer* untuk *Java* yang tersedia sebagai *plugin* pada *IDE Eclipse*. Dengan *WindowBuilder*, dapat merancang tampilan *GUI* secara visual menggunakan metode *drag-and-drop*, tanpa harus menulis semua kode secara manual. *WindowBuilder* mendukung pengembangan antarmuka menggunakan *Swing*, *SWT*, maupun *GWT*.

Berikut beberapa fungsi dari *WindowBuilder*:

1. Sinkronisasi otomatis antara editor visual dan kode.
2. Editor visual untuk mendesain layout.
3. Mempercepat proses pembuatan antarmuka *Java*

2.1.2 Java Swing

Java Swing adalah *library GUI* yang termasuk dalam *Java Foundation Classes (JFC)*. *Swing* digunakan untuk membuat aplikasi desktop dengan tampilan yang dapat dikustomisasi. *Library* ini menyediakan banyak komponen *GUI*, mulai dari komponen dasar seperti tombol hingga komponen kompleks seperti *tabel* dan *dialogs*.

Keunggulan *Java Swing*:

1. Bersifat *lightweight* (tidak bergantung pada sistem operasi).
2. Memiliki banyak komponen bawaan.
3. Memungkinkan pembuatan *layout* yang fleksibel.

2.1.3 Container J-Frame

JFrame adalah kelas utama dalam *swing* yang digunakan untuk membuat jendela aplikasi. *JFrame* berfungsi sebagai wadah (*container*) utama yang menampung berbagai komponen *GUI* seperti tombol, *label*, dan *text field*.

2.1.4 Component GUI Java Swing

Komponen *GUI* adalah elemen visual yang digunakan pengguna untuk berinteraksi dengan program. Beberapa komponen dasar yang digunakan pada praktikum ini antara lain:

1. *JLabel*

Digunakan untuk menampilkan teks atau informasi statis.
Fungsi: memberi petunjuk, judul, atau deskripsi *input*.

2. *TextField*

Komponen *input* untuk menerima masukan berupa teks dari pengguna. Biasanya digunakan untuk formulir, *input* nilai, atau data sederhana.

3. *Jbutton*

Tombol yang dapat ditekan pengguna untuk melakukan aksi tertentu. Umumnya digunakan bersama *ActionListener*.

4. *JPanel*

Wadah tambahan yang digunakan untuk menata komponen secara rapi berdasarkan layout tertentu. Membantu modularisasi tampilan aplikasi.

5. *JComboBox*

Drop-down menu untuk memilih salah satu opsi dari daftar yang dibuat.

6. *JCheckBox* & *JRadioButton*

Komponen input pilihan:

- i. *JCheckBox* (pilihan bisa lebih dari satu)
- ii. *JRadioButton* (pilihan tunggal dalam satu grup)

2.1.5 *Event Handling*

Event handling adalah mekanisme untuk menangani aksi pengguna, seperti menekan tombol, mengetik teks, atau memilih opsi. Dalam *Java Swing*, *event handling* dilakukan menggunakan *listener*, seperti:

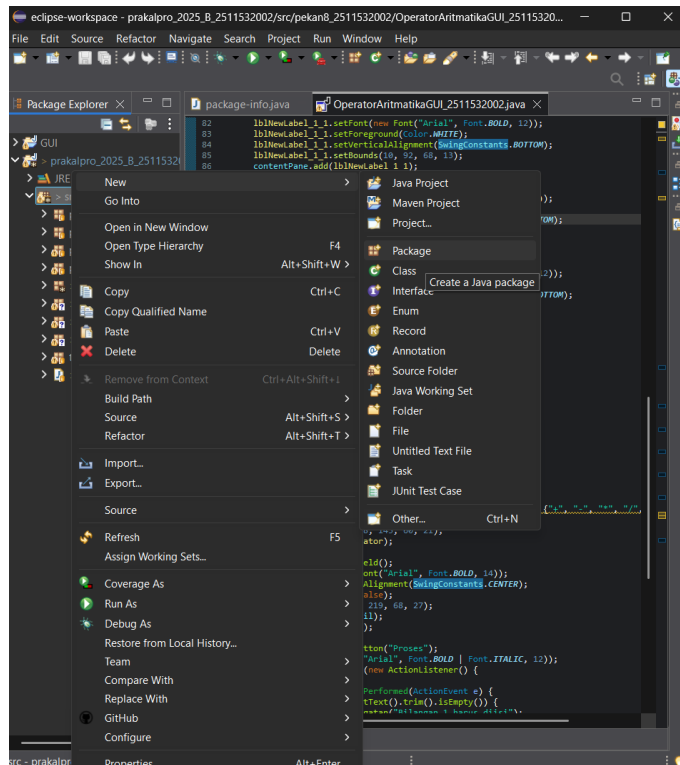
- i. *ActionListener*, menangani aksi tombol
- ii. *KeyListener*, menangani *input keyboard*.
- iii. *MouseListener*, menangani klik *mouse*.

Event handling memastikan aplikasi merespon interaksi pengguna dengan benar dan dinamis.

2.2 Langkah Kerja

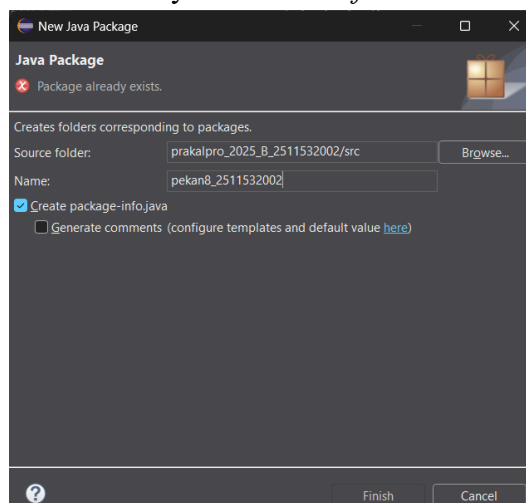
2.2.1 Pembuatan *Package* dan *Class*

- a) Sebelum membuat suatu program *user* perlu untuk membuat *package* terlebih dahulu. Klik kanan pada *src* lalu pilih “New” dan klik “Package”.



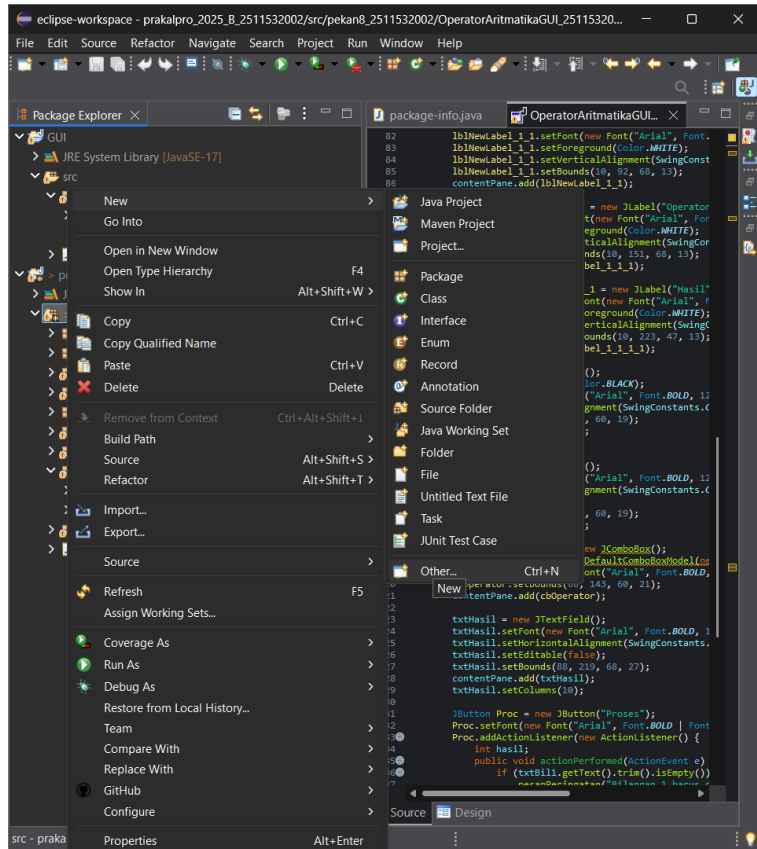
2.2.1.1 Pembuatan *Package* pekan8_2511532002

- b) Setelah itu akan muncul *pop up* “Java Package”, buat nama *package* dengan ketentuan tanpa *capslock*, *space*, dan karakter khusus lainnya. Lalu klik “finish”.



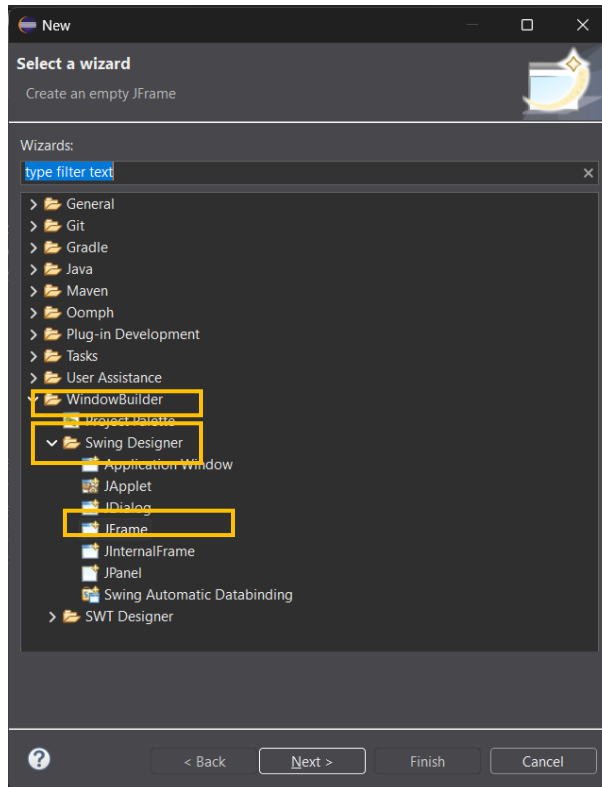
2.2.1.2 Pembuatan *Package* pekan8_2511532002

- c) Selanjutnya adalah membuat membuat *class*. pada package yang telah dibuat tapi klik kanan lalu pilih “New” dan klik “Other”.



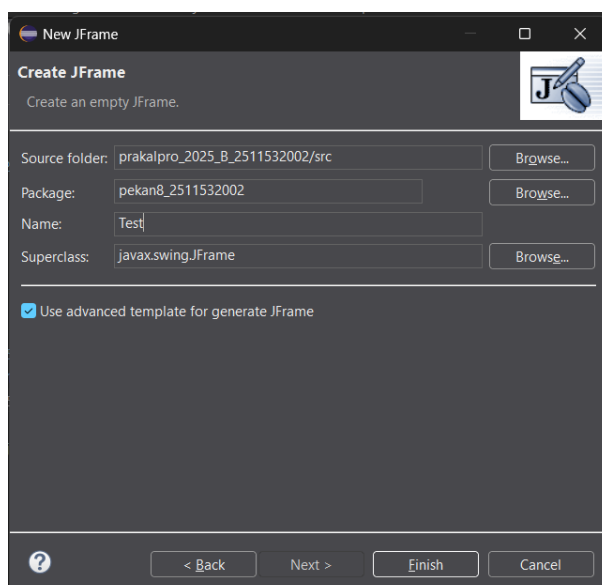
2.2.1.3 Pembuatan Class OperatorAritmatikaGUI_2511532002

- d) Setelah itu akan muncul menu *pop up* “Selected a Wizard”, lalu pilih *Window Builder* dan *Swing Designer*. Pada menu *swing designer* pilih *JFrame*. Lalu klik “Next”.



2.2.1. 4 Pembuatan Class *OperatorAritmatikaGUI_2511532002*

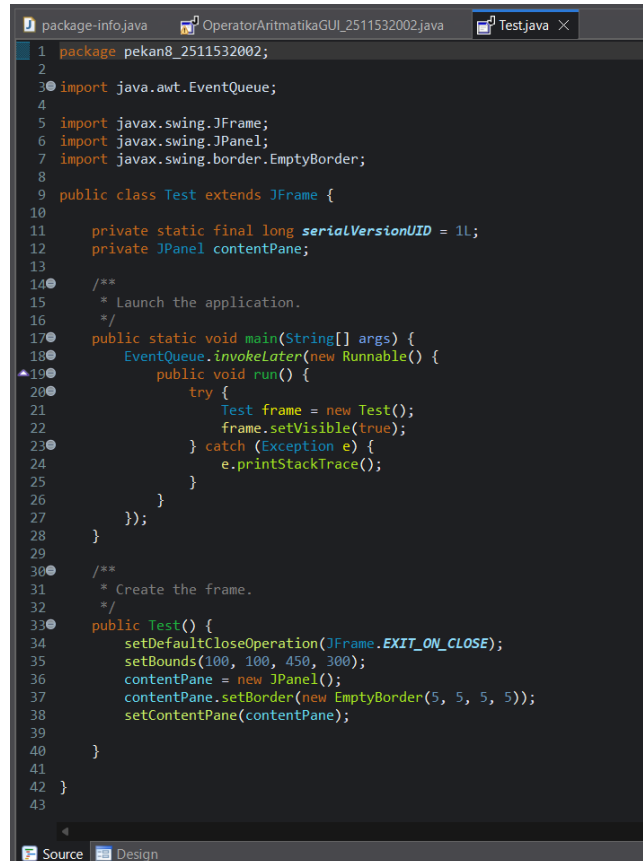
Setelah itu pada menu “Create JFrame” buat nama *class*, dengan ketentuan nama harus *capslock* diawal kalimat dan



2.2.1. 5 Pembuatan Class *OperatorAritmatikaGUI_2511532002*

tanpa spasi. Setelah itu aktifkan “Use advanced template for generate JFrame”. Dan klik “Finish”.

Maka akan kelas baru, dengan tab “Source” dan “Design”.

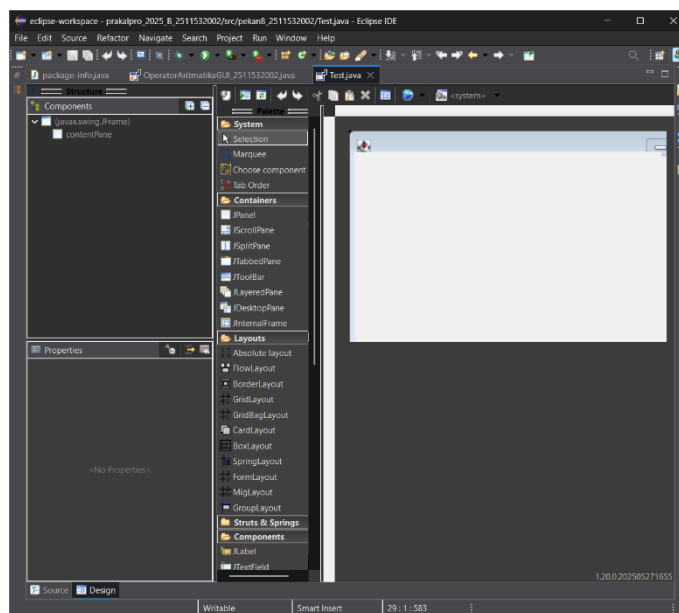


```

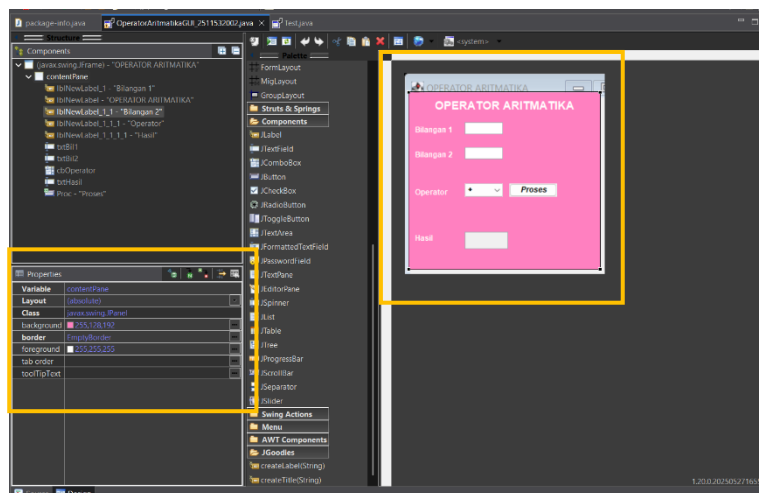
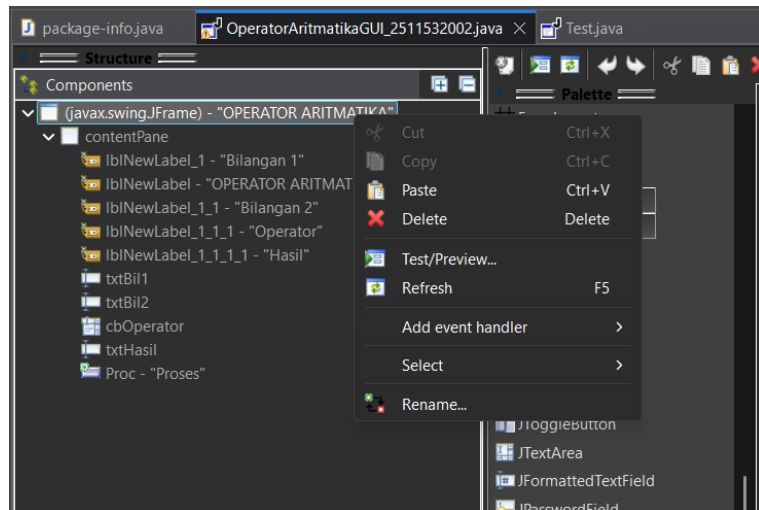
1 package pekan8_2511532002;
2
3 import java.awt.EventQueue;
4
5 import javax.swing.JFrame;
6 import javax.swing.JPanel;
7 import javax.swing.border.EmptyBorder;
8
9 public class Test extends JFrame {
10
11     private static final long serialVersionUID = 1L;
12     private JPanel contentPane;
13
14     /**
15      * Launch the application.
16      */
17     public static void main(String[] args) {
18         EventQueue.invokeLater(new Runnable() {
19             public void run() {
20                 try {
21                     Test frame = new Test();
22                     frame.setVisible(true);
23                 } catch (Exception e) {
24                     e.printStackTrace();
25                 }
26             }
27         });
28     }
29
30     /**
31      * Create the frame.
32      */
33     public Test() {
34         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
35         setBounds(100, 100, 450, 300);
36         contentPane = new JPanel();
37         contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
38         setContentPane(contentPane);
39     }
40 }
41
42 }
43

```

2.2.1. 7 Source Class OperatorAritmatikaGUI_2511532002



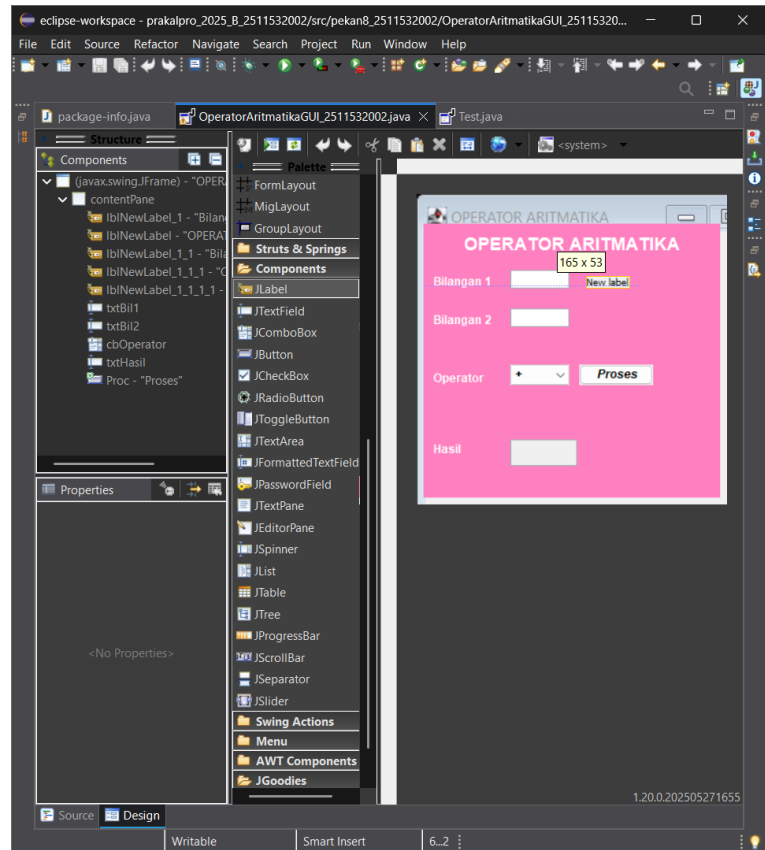
2.2.1. 6 Tab Design Class OperatorAritmatikaGUI_2511532002



- 1) Setelah menyiapkan panel, *user* dapat mulai mendesain tampilan.
- 2) Disini saya memulai dengan menamhakan "*JLabel*",caranya pada "*Palette*", pilih *component*

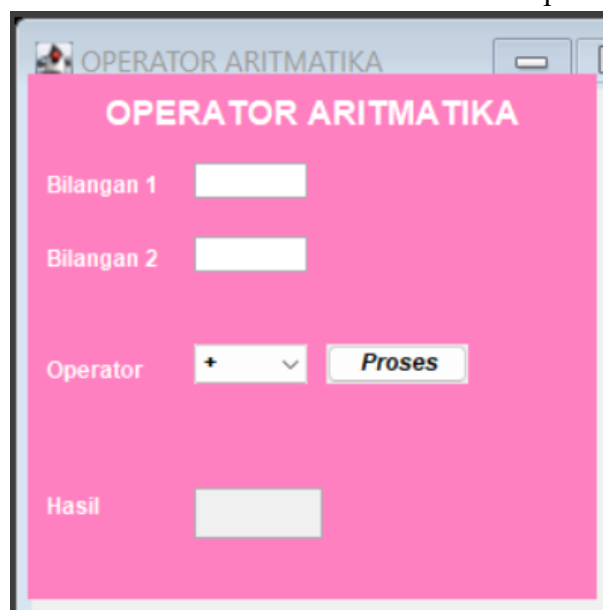
- 2) Disini saya memulai dengan menamhakan “JLable”,caranya pada “Palette”, pilih *component*

“Jlable” drag and drop ke panel. Maka akan muncul text box “New Lable”.



2.2.1. 10 Penambahan Component

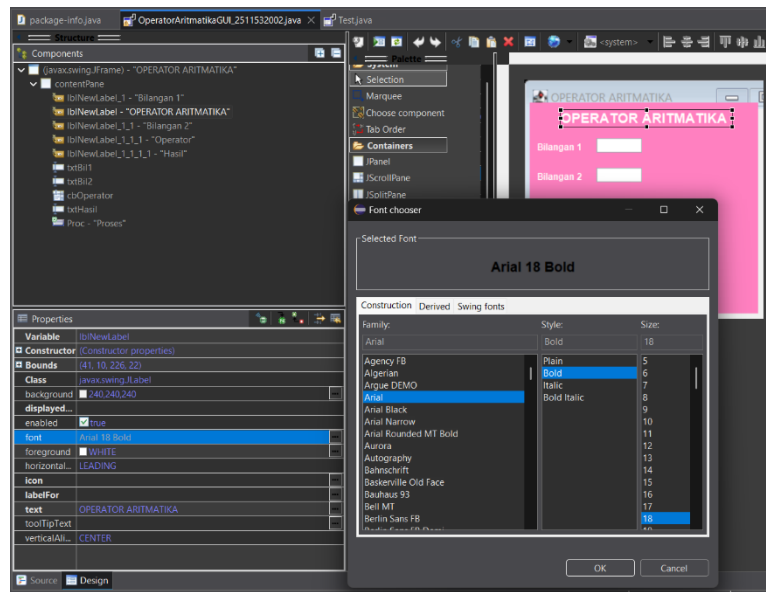
- 3) Selanjutnya edit text box “JLabel” untuk judul “OPERATOR ARITMATIKA”. Dan komponen lain



2.2.1. 11 Penambahan Component JLabel

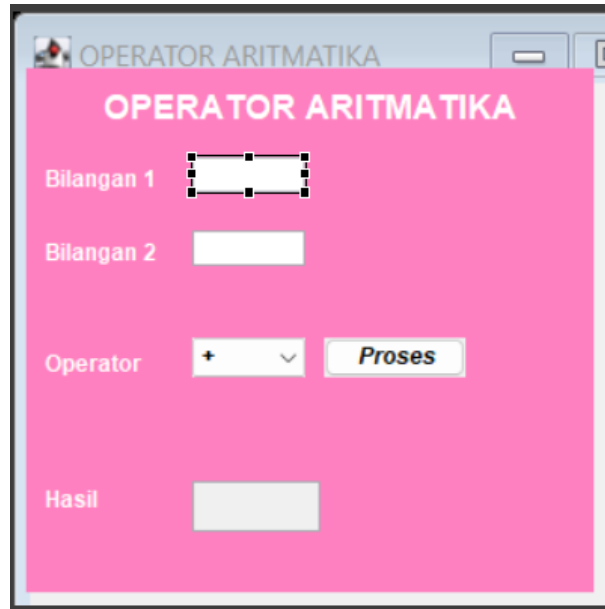
seperti “*Bilangan 1*”, “*Bilangan 2*”, “*Operator*”, dan “*Hasil*”.

- 4) *User* dapat mengatur property dari tiap komponen, seperti memilih *font*, *alignment*, warna dan lainnya. Misalkan disini saya mengedit *component* “*OPERATOR ARITMATIKA*”, mengganti *font* nya menjadi *Arial* dengan *style Bold* dan ukuran *18*.



2.2.1. 12 Pengeditan Component

- 5) Selanjutnya *component* bertipe *JTextFiled* untuk menerima *input* atau memberikan *output*. Seperti berikut:



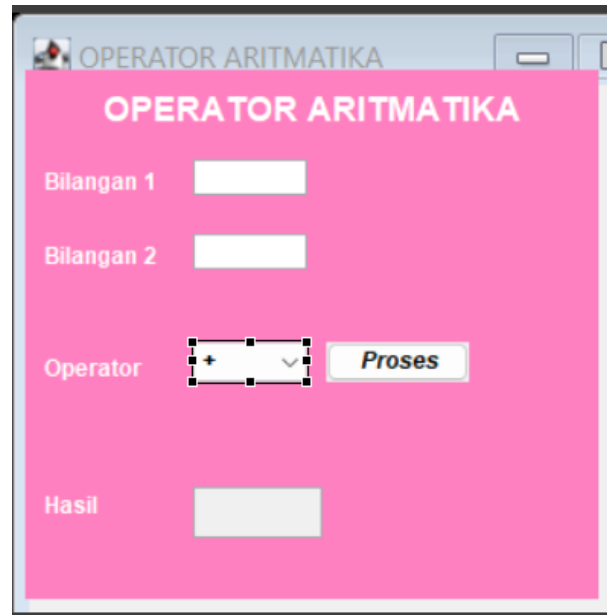
2.2.1. 13 Penambahan Component *JTextField*

Tambahkan tiga sebuah *text field*:

- txtBil1* untuk *input* bilangan pertama
- txtBil2* untuk *input* bilangan kedua
- txtHasil* untuk menampilkan hasil operasi
Set *editable* pada *txtHasil* menjadi *false* agar tidak dapat diedit manual.



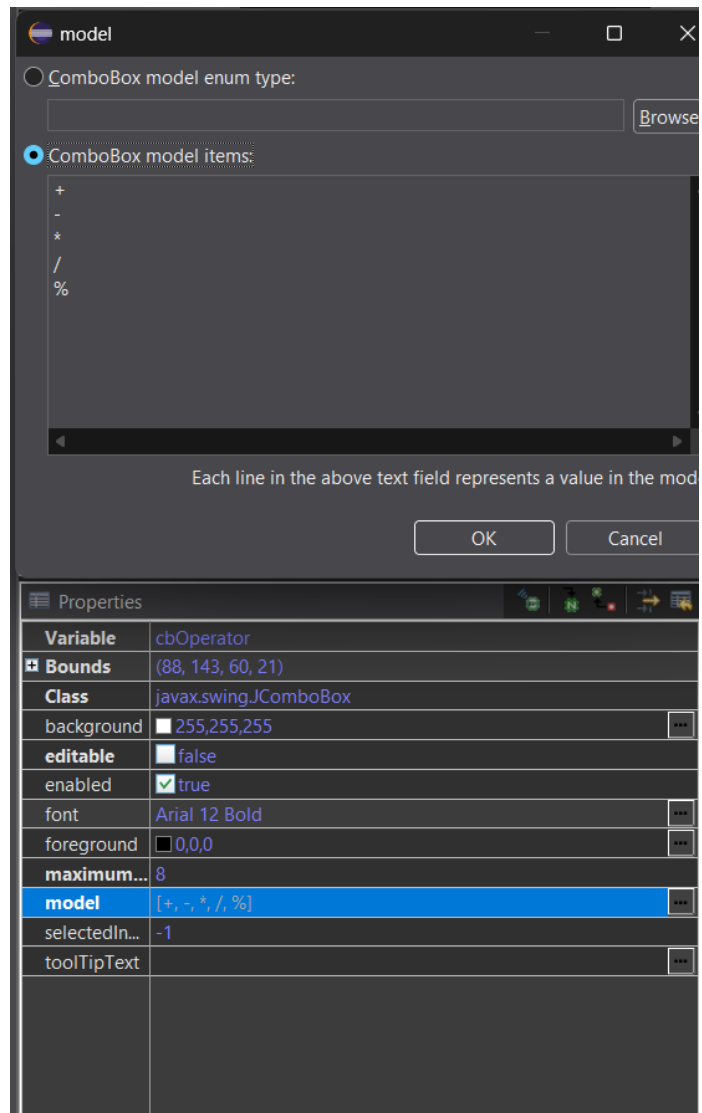
- 6) Lalu component *JComboBox* untuk pilihan operator.



2.2.1. 14 Penambahan Component *JComboBox*

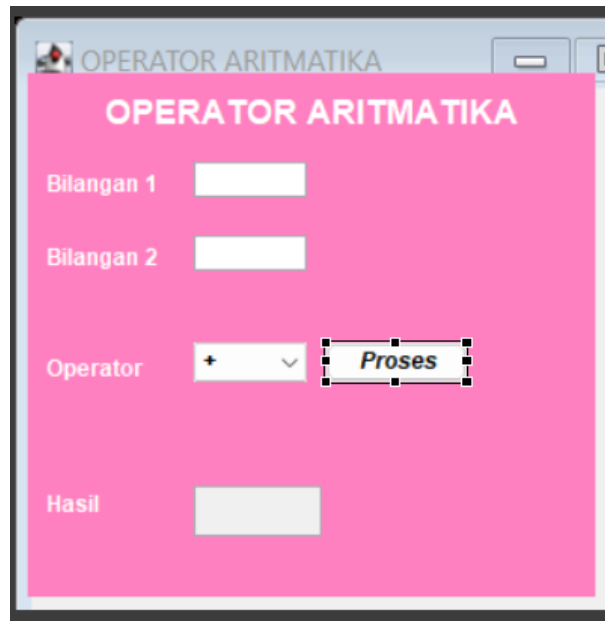
Disini saya mengisi dengan pilihan + (penjumlahan), - (pengurangan), * (perkalian), / (pembagian), dan % (untuk mod atau sisa hasil bagi).

Dari sini programmer dapat memasukan dan megedit operasi yang akan digunakan nantinya.



2.2.1. 15 Mengedit pilihan/ operasi JComboBox

- 7) Terakhir, agar bilangan dapat di kalkulasikan, perlu menggunakan *Jbutton* untuk tombol proses.



2.2.1. 16 Penambahan Component *Jbutton* Proses

- h) Selanjutnya menginisiasi variabel komponen yang digunakan. Deklarasi variabel pada bagian atas *class*, agar komponen dapat diakses oleh seluruh *method* dalam *class*.

```
public class OperatorAritmatikaGUI_2511532002 extends JFrame {

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JTextField txtBil1;
    private JTextField txtBil2;
    private JTextField txtHasil;
```

- i) Membuat *methode* tambahan untuk menampilkan pesan peringatan dan pesan kesalahan, dimana *method* ini digunakan untuk validasi *input* dan menampilkan pesan jika terjadi *error*.

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

- j) Selanjutnya menambahkan fungsi proses pada tombol proses. Pada tombol Proses, tambahkan *event handling* menggunakan *ActionListener*, sesuai dengan proses yang dipilih oleh *user*.

```

JButton Proc = new JButton("Proses");
Proc.setFont(new Font("Arial", Font.BOLD | Font.ITALIC, 12));
Proc.addActionListener(new ActionListener() {
    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 2 harus diisi");
        } else {
            try {
                int a = Integer.parseInt(txtBil1.getText());
                int b = Integer.parseInt(txtBil2.getText());
                int c = cbOperator.getSelectedIndex();
                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) {
                pesanError("Bilangan 1 dan 2 harus angka");
            }
        }
        txtHasil.setText(String.valueOf(hasil));
    }
});
Proc.setBounds(158, 143, 75, 21);
contentPane.add(Proc);

```

2.2.1. 17 Syntax JButton atau Tombol Proses

Berikut merupakan penjelasan *syntax tombol proses* lebih lanjut:

1) *JButton Proc = new JButton("Proses");*

Membuat objek tombol dengan label "Proses" yang akan muncul pada *GUI*

2) *Proc.addActionListener(new ActionListener() {*

Memberi aksi pada tombol saat diklik. Setiap kali tombol ditekan, method *actionPerformed()* akan dijalankan.

3) *int hasil;*

Variabel ini digunakan untuk menyimpan hasil perhitungan.

4) *public void actionPerformed(ActionEvent e) {*

Semua logika yang dijalankan ketika tombol ditekan berada di dalam method ini.

5) *if (txtBil1.getText().trim().isEmpty()) {*

pesanPeringatan("Bilangan 1 harus diisi");

Mengecek apakah bilangan 1 kosong, jika iya maka akan menampilkan pesan peringatan “Bilangan 1 harus diisi”.

6) *else if (txtBil2.getText().trim().isEmpty()) {*

pesanPeringatan("Bilangan 2 harus diisi");

Jika bilangan 2 kosong, maka muncul pesan peringatan “Bilangan 2 harus diisi”.

7) *int a = Integer.parseInt(txtBil1.getText());*

int b = Integer.parseInt(txtBil2.getText());

Mengubah nilai teks menjadi angka *integer*.

8) *int c = cbOperator.getSelectedIndex();*

9) Mengambil index dari operator pada *ComboBox*

10) Melakukan oprasi perhitungan.

```
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;
}
```

11) }catch (NumberFormatException ex) {

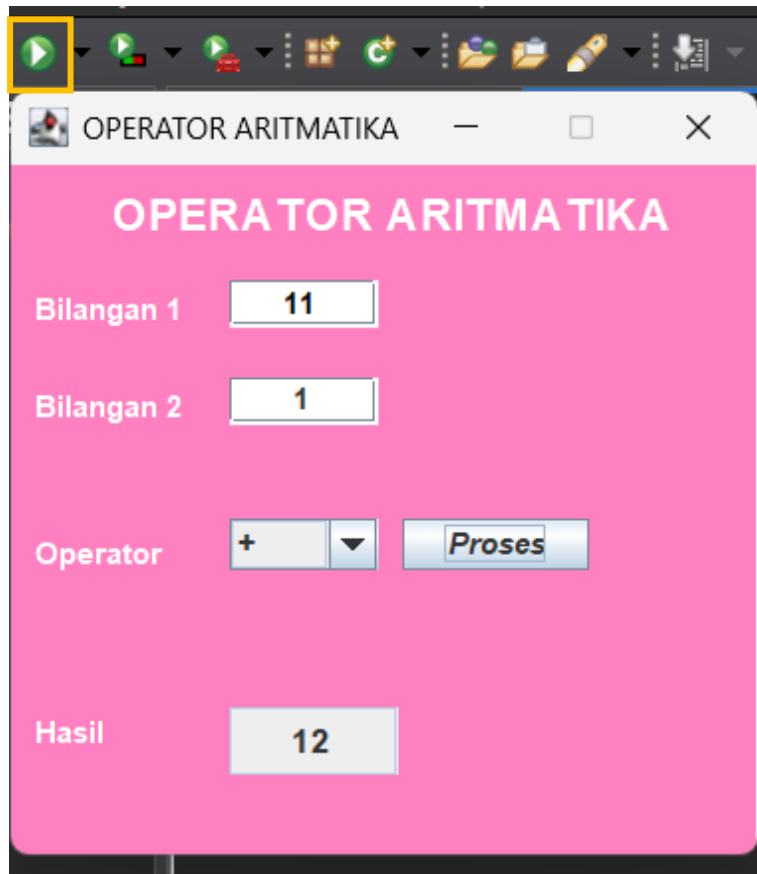
pesanError("Bilangan 1 dan 2 harus angka");

Memastikan *user input* angka, Jika *user* memasukkan huruf, simbol, atau karakter *non-numeric*, maka akan muncul pesan eoror "Bilangan 1 dan 2 harus angka".

12) *txtHasil.setText(String.valueOf(hasil));*

13) Menampilkan hasil ke *TextField*, dan mengubah angka menjadi *String*.

- k) Setelah itu maka, *user* sudah dapat menguji program dengan menjalankannya, pada *toolbar* klik tombol “Run”, maka akan muncul *pop up* panel proram.



2.2.1. 18 Hasil Uji Coba Program

Disini dapat dilihat bahwa *program* “*OPERATOR ARITMATIKA*” telah berjalan. Berikut hasil analisis dari uji coba lainnya:

a) Uji Coba 1: Penjumlahan



The screenshot shows a window titled "OPERATOR ARITMATIKA". Inside, the title "OPERATOR ARITMATIKA" is displayed in pink. Below it, there are four input fields: "Bilangan 1" with the value "11", "Bilangan 2" with the value "1", "Operator" with a dropdown menu showing "+", and "Hasil" with the value "12". A "Proses" button is located next to the operator dropdown.

2.2.1. 19 Hasil Uji Coba 1

b) Uji Coba 2: Pengurangan



The screenshot shows the same window titled "OPERATOR ARITMATIKA". The input fields are: "Bilangan 1" with the value "11", "Bilangan 2" with the value "1", "Operator" with a dropdown menu showing "-", and "Hasil" with the value "10". The "Proses" button is still present next to the operator dropdown.

2.2.1. 20 Hasil Uji Coba 2

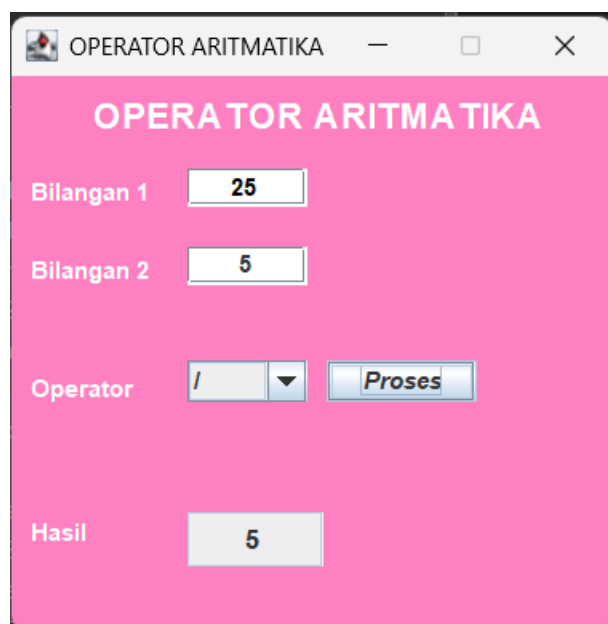
c) Uji Coba 3: Perkalian



The screenshot shows a window titled "OPERATOR ARITMATIKA". Inside, the title "OPERATOR ARITMATIKA" is displayed in large pink letters. Below this, there are four input fields: "Bilangan 1" with the value "11", "Bilangan 2" with the value "2", "Operator" with a dropdown menu showing "*", and "Hasil" with the value "22". A blue button labeled "Proses" is located to the right of the operator dropdown.

2.2.1. 21 Hasil Uji Coba 3

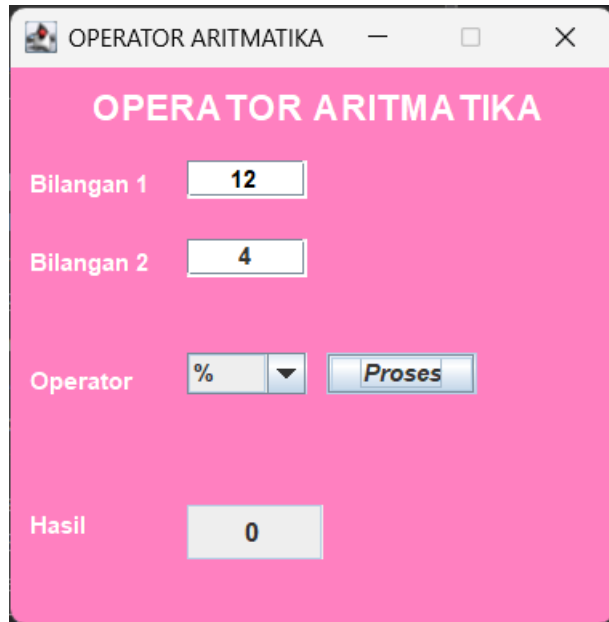
d) Uji Coba 4: Pembagian



The screenshot shows the same "OPERATOR ARITMATIKA" window. The input fields now contain: "Bilangan 1" with the value "25", "Bilangan 2" with the value "5", "Operator" with a dropdown menu showing "/", and "Hasil" with the value "5". The "Proses" button remains to the right of the operator dropdown.

2.2.1. 22 Hasil Uji Coba 4

e) Uji Coba 5: Hasil Sisa Bagi / $Mod(\%)$



OPERATOR ARITMATIKA

OPERATOR ARITMATIKA

Bilangan 1 12

Bilangan 2 4

Operator % ▾ Proses

Hasil 0

2.2.1. 24 Hasil Uji Coba 5, Kondisi Pertama



OPERATOR ARITMATIKA

OPERATOR ARITMATIKA

Bilangan 1 12

Bilangan 2 5

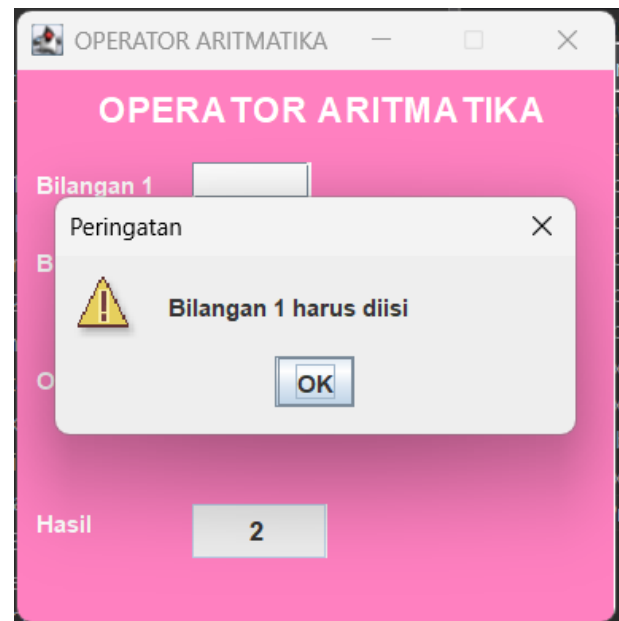
Operator % ▾ Proses

Hasil 2

2.2.1. 23 Hasil Uji Coba 5, Kondisi Kedua

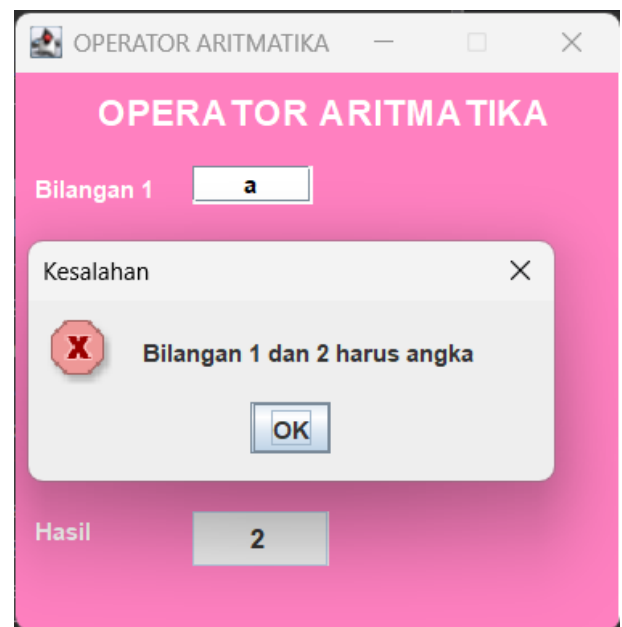
f) Uji Coba 6: Eror

- Bilangan tidak diisi.



2.2.1. 25 Hasil Uji Coba 6, Bilangan 1 kosong

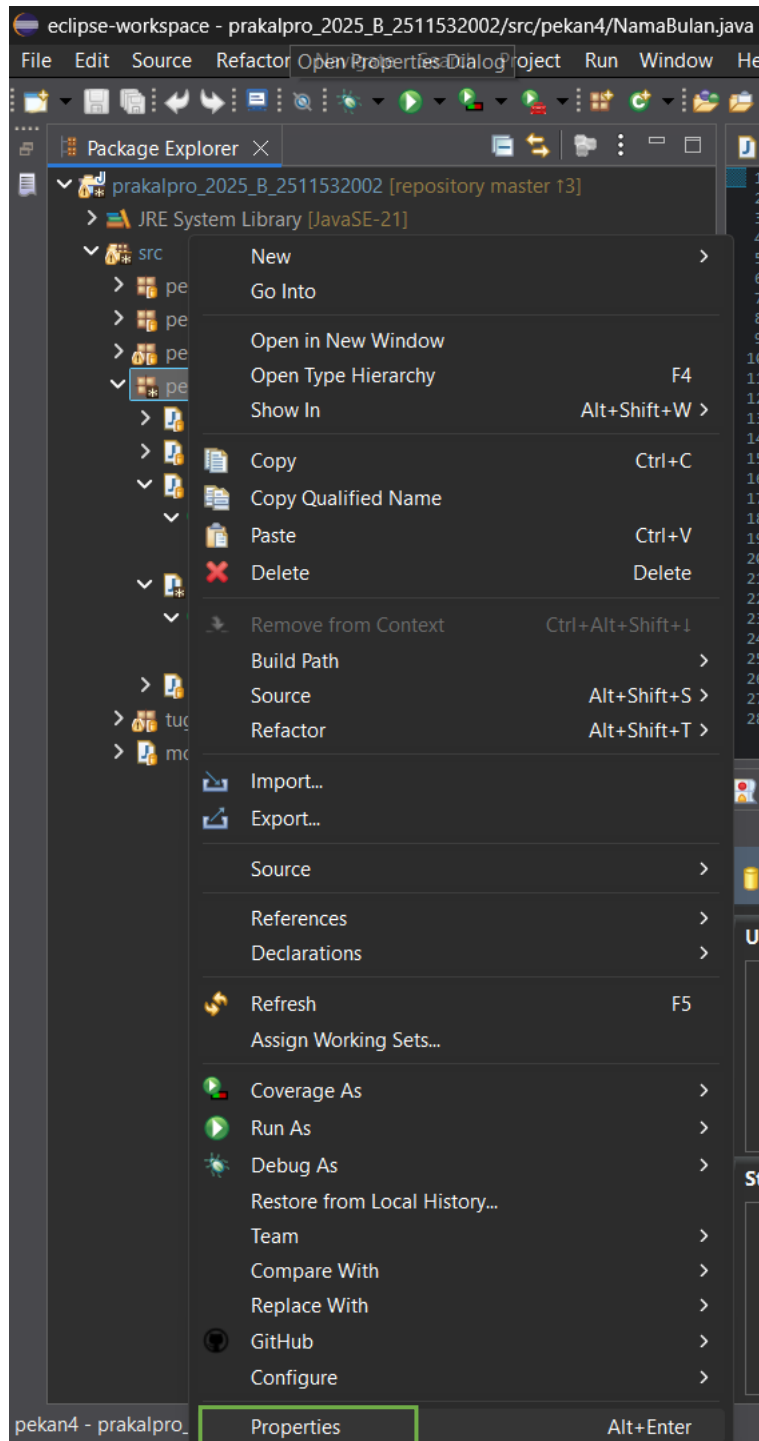
- User input bilangan non-numeric.



2.2.1. 26 Hasil Uji Coba 6, Bilangan 1 tidak berupa numerik

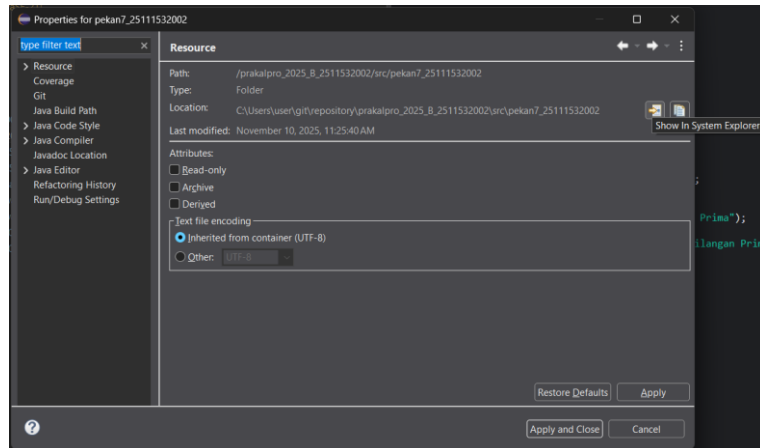
2.2.2 Upload ke GitHub

1. Setelah semua program selesai dibuat, selanjutnya adalah *user* perlu untuk memasukkan program yang dibuat di Eclipse ke *repository* GitHub. Selain dengan cara “*Team*” atau “*Commit and Push*”. Bisa juga dengan cara *upload* manual. Caranya klik kanan pada *package*, lalu pilih *properties*.



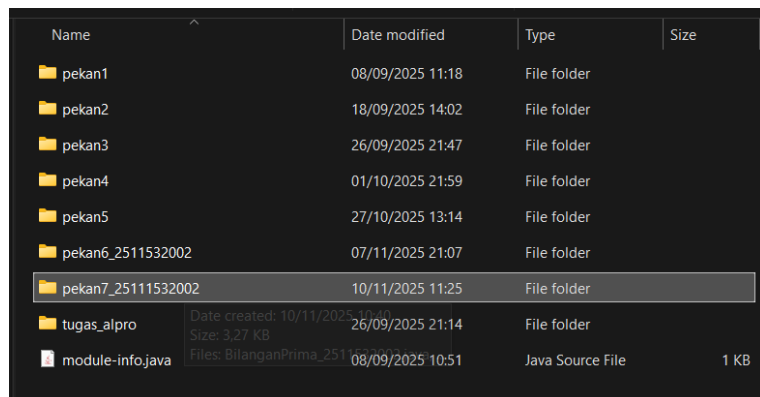
2.2.1 Package Properties of pekan7_2511532002

2. Maka akan muncul *pop up menu* “*Properteis for pekan8_2511532002*”. Lalu klik *icon* “*Show in System Folder*”.



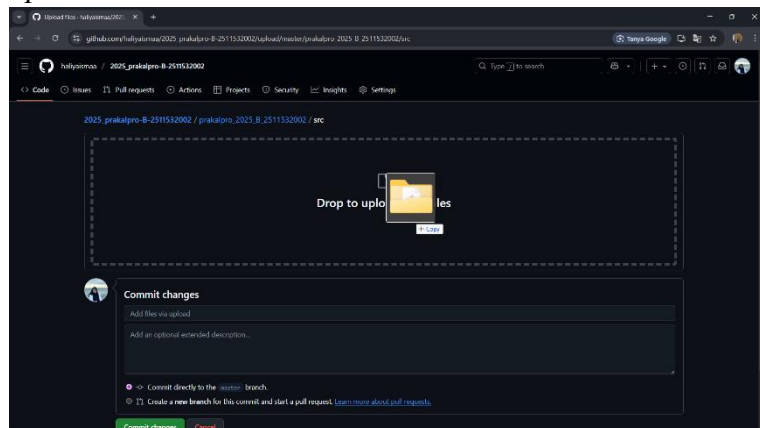
2.2 2 Properteis for pekan8_2511532002

3. Maka akan menunjukan lokasi *file* “*pekan8_2511532002*” pada *File Explorer computer*.



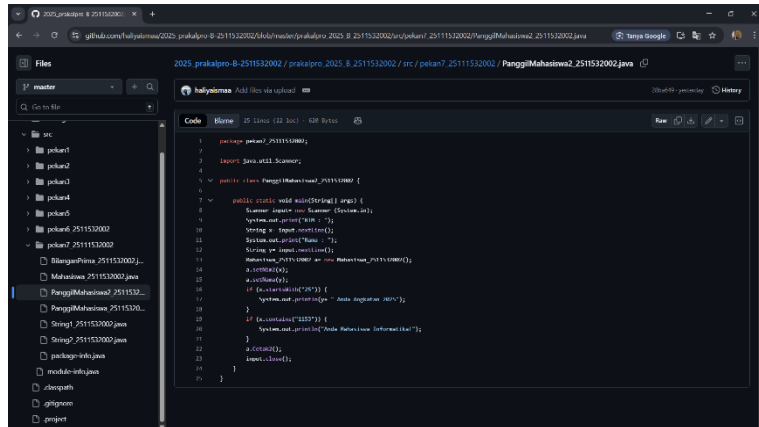
2.2 3 File Location of pekan8_2511532002

4. Setelah itu *drag and drop* file “*pekan8_2511532002*” ke GitHub. Lalu tambahkan keterangan jika perlu. Setelah itu klik *upload*.



2.2 4 Drag and Drop file pekan8_2511532002 ke GitHub

5. Maka dapat dilihat pada *repository GitHub* bahwa file “peka8_2511532002” telah terunggah via *Upload*.



2.2.5 Hasil Upload Package pekan8_2511532002 ke GitHub

Dapat kita lihat disini jika kita buka, maka akan muncul *file package* dan *class* yang telah di buat di Eclipse.

BAB III

KESIMPULAN

3.1 Kesimpulan

Berdasarkan praktikum pekan ke-8 dengan topik “Penerapan *GUI* dalam Pemrograman *Java*” melalui program “*Operasi Aritmatika*”, dapat disimpulkan bahwa penggunaan *Java Swing* dan *WindowBuilder* mempermudah pogramer dalam pembuatan antarmuka aplikasi. Penyusunan komponen seperti *JLabel*, *JTextField*, *JComboBox*, dan *JButton* dapat dilakukan secara visual, sehingga mempermudah penataan *layout* tanpa perlu menuliskan kode secara manual.

Program berjalan dengan baik, di mana *user* dapat memasukkan angka, memilih operasi, dan memperoleh hasil perhitungan secara otomatis. Validasi *input* juga berfungsi, ditunjukkan dengan munculnya peringatan saat pengguna memasukkan nilai *non-numerik* atau disaat *user* tidak memasaukan bilangan 1 dan 2. Secara keseluruhan, praktikum ini membantu memahami dasar pembuatan *GUI* di *Java* serta penerapannya dalam pembuatan aplikasi sederhana yang fungsional dan *user-friendly*.

3.2 Saran

Saran untuk praktikum selanjutnya adalah agar mahasiswa lebih teliti dalam memahami penulisan *component* yang tepat sesuai dengan kebutuhan program, terutama saat mengatur komponen *GUI* dan menangani input pengguna. Selain itu, penting untuk membedakan dengan benar antara tipe data *String* dan *numerik* agar tidak terjadi kesalahan logika pada proses perhitungan maupun tampilan *output*.

DAFTAR PUSTAKA

- [1] Eclipse Foundation, “WindowBuilder – Eclipse Project.” [Daring]. Tersedia pada: <https://eclipse.dev/windowbuilder/>. [Diakses: 11-Nov-2025].
- [2] E. H. Chua, “Java GUI Programming – Swing (JFC).” [Daring]. Tersedia pada: https://www3.ntu.edu.sg/home/ehchua/programming/java/J4a_GUI.html. [Diakses: 11-Nov-2025].
- [3] GeeksforGeeks, “Introduction to Java Swing.” [Daring]. Tersedia pada: <https://www.geeksforgeeks.org/java/introduction-to-java-swing/>. [Diakses: 11-Nov-2025].