LAPORAN PRAKTIKUM

PEMROGRAMAN ALGORITMA DAN PEMROGRAMAN

“PROGRAM GUI 1: APLIKASI OPERATOR ARITMATIKA MENGGUNAKAN WINDOWBUILDER PADA ECLIPSE IDE”

Disusun Oleh :

Dzhillan Dzhalila

2511531001

Dosen Pengampu:

Dr. Wahyudi, S.T, M.T

Asisten Praktikum:

Aufan Taufiqurrahman

FAKULTAS TEKNOLOGI INFORMASI

DEPARTEMEN INFORMATIKA

UNIVERSITAS ANDALAS

2025

KATA PENGANTAR

Puji syukur penulis panjatkan atas kehadirat Allah SWT Tuhan Yang Maha Esa yang telah memberikan rahmat dan berkatnya, sehingga penulis dapat menyelesaikan laporan praktikum Algoritma Pemrograman dan Pemrograman dengan baik. Laporan ini disusun sebagai dokumentasi dan refleksi atas praktikum yang telah dilaksanakan pada pertemuan minggu ke-8 untuk mata kuliah Praktikum Algoritma Pemrograman. Praktikum ini berfokus pada pemahaman dan pengaplikasian *plugin* WindowBuilder untuk membangun aplikasi GUI pada Eclipse Java.

Melalui praktikum ini, hasil yang diharapkan adalah penulis dapat memahami dan mengaplikasikan *plugin* WindowBuilder dengan baik dan efisien. Melalui projek sederhana yang dilakukan pada praktikum ini membantu penulis untuk mengembangkan aplikasi GUI dengan Bahasa Java sebagai bahasa pemrograman.

Penulis menyadari bahwa, laporan praktikum ini masih memiliki banyak kekurangan. Oleh karena itu, kritik dan saran yang membangun sangat diharapkan demi perbaikan dan pengembangan di masa mendatang. Semoga laporan ini dapat memberikan manfaat, baik sebagai bahan pembelajaran penulis maupun referensi pembaca dalam memahami dasar-dasar pemrograman komputer.

Padang, 2025

Dzhillan Dzhalila

251531001

DAFTAR ISI

BAB I
PENDAHULUAN

1.1 Latar Belakang

Dalam pengembangan aplikasi modern, antarmuka grafis (*graphic interface*) merupakan salah satu aspek yang menentukan keefektifan dari suatu aplikasi. Pengalaman pengguna yang positif sangat bergantung pada kualitas UI (*User Interface*). Untuk memudahkan pengembangan aspek visual, menggunakan GUI adalah pilihan yang tepat. GUI atau *Graphical User Interface* memberikan kemudahan melalui penggunaan elemen visual seperti ikon,tombol atau label yang memungkinkan interaksi antara pengguna dan sistem.

Java menyediakan *tools* yang mendukung pengembangan GUI, salah satunya adalah *ekstention* WindowBuilder. *Plugin* ini memungkinkan kita untuk membuat tata letak aplikasi GUI secara visual tanpa perlu menulis kode *layout* secara manual. WindowBuilder dapat membaca dan menulis hampir semua format dan merekayasa baik sebagian besar kode GUI Java yang ditulis tangan. WindowBuilder juga mendukung pengeditan kode bentuk bebas bagi pengguna yang ingin memindahkan, mengganti nama, bahkan mengganti kode kode yang ada. Bisa dikatakan bahwa, WindowBuilder merupakan *designer* GUI Java dua arah yang kuat dan mudah digunakan.

1.2 Tujuan

1. Memahami konsep dasar seperti komponen-komponen utama GUI pada Java dan alur eksekusi program berbasis GUI.
2. Menguasai penggunaan WindowBuilder untuk mendesain antarmuka secara visual (*drag – and – drop*).
3. Mengimplementasikan logika program dalam GUI dalam merancang dan menulis kode fungsionalitas sederhana seperti aplikasi Operator Aritmatika dll.

1.3 Manfaat

1. Memberikan penguasaan akan pemahaman framework GUI melalui WindowBuilder sehingga dapat menjadi fondasi awal untuk memahami GUI modern lainnya.
2. Meningkatkan pemahaman akan konsep *event-driven programming* dan *object-oriented design* dalam konteks nyata.

BAB II
PEMBAHASAN

2.1 Dasar Teori

GUI (*Graphical User Interface*) adalah sistem antarmuka sistem operasi berbasis grafis seperti ikon, tombol, menu dan representasi visual lainnya yang mendukung interaksi pengguna dengan sistem. Sebelum adanya GUI, perintah harus diketik dari *keyboard* untuk mendapat respons atau output dari komputer yang juga disebut dengan *text-based interfaces*. Sementara input GUI memanfaatkan pointer seperti *mouse* yang akan memudahkan pengguna untuk mengakses sebuah program. Dalam pengembangan aplikasi GUI, kita dapat menggunakan Java untuk menjadi bahasa pemrograman aplikasi tersebut. Java sudah menyediakan serangkaian komponen antarmuka pengguna yang dapat digunakan untuk membangun aplikasi GUI.

Pada Eclipse IDE , WindowBuilder adalah salah satu *plugin* yang dapat digunakan untuk membuat formulir JFC/Swing atau AWT (Abstract Window Toolkit) . Terdapat banyak sekali formulir GUI yang tersedia di IDE. Masing masing memiliki kegunaan yang berbeda dalam hal waktu desain, tampilan proses formulie dan kode yang dihasilkan untuk kelas formulir tersebut. Beberapa templatenya adalah JApplet, JDialog, JFrame, JInternalFrame, JPanel dll. Dan pada praktikum kali ini, kita menggunakan template JFrame untuk template formulir aplikasi GUI yang akan kita buat. Formulir JFrame merupakan jendela aplikasi tingkat atas yang akan menjadi wadah untuk *basic controls* atau komponen aplikasi antarmuka seperti Jbutton, JTextField, JLabel dll.

Aplikasi GUI di Java bersifat *event driven* yang berarti saat *user* berinteraksi dengan komponen GUI, interaksi tersebut disebut dengan *event* yang akan memicu program untuk melaksanakan sebuah proses. Kode program yang akan dijalankan pada saat *event* terdeteksi dikenal sebagai *event handler*. Keseluruhan proses penggunaan *event* disebut dengan *event handling*. Dalam penanganan *event* ada empat bagian penting yang harus diketahui

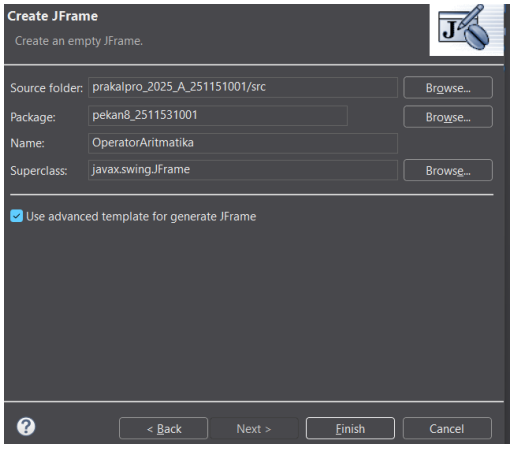
1. *Event Object* : objek yang mendiskripsikan sebuah event yang *ditrigger* oleh sumber *event.*
2. *Event Handler* : metode yang menerima objek event dan melakukan respond yang sesuai dengan objek event tersebut.
3. *Event Listener/Handler :* antarmuka yang akan menangani event yang terjadi. Setiap jenis event memiliki antarmuka yang bersesuaian dan *Listener* tersebut harus diimplementasikan oleh kelas yang aka menangani event.
4. *Event Source :* pembangkit sebuah objek event.

2.2 Program Aplikasi Operator Aritmatika

Untuk membuat aplikasi Operator Aritmatika kita menggunakan *plugin* WindowBuilder dan memilih *Swing Designer* sebagai alat bantu GUI builder dan membuat jendela dengan memilih salah satu template formulir pada *Swing Designer* yaitu adalah JFrame. Dan *superclass* program ini akan langsung menjadi "javax.swing.JFrame".

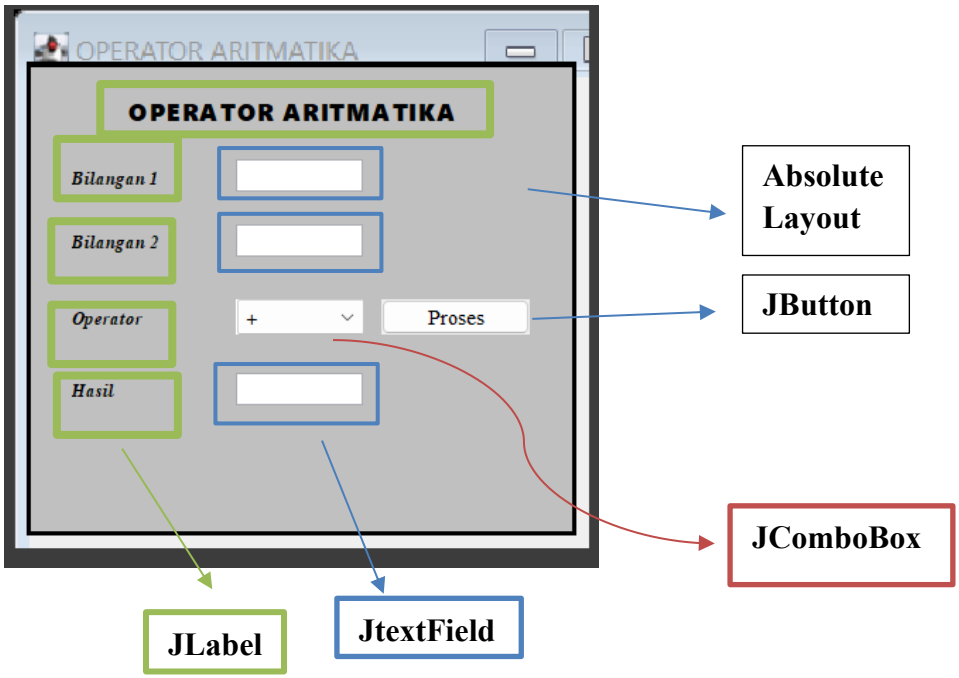
Adapun langkah-langkah praktis untuk membuat GUI dari aplikasi Operasi Aritmatika ini adalah sebagai berikut :

1. Membuat proyek baru dengan nama proyek “OperasiAritmatika”.



Gambar 2.1 : Proyek Operator Aritmatika.

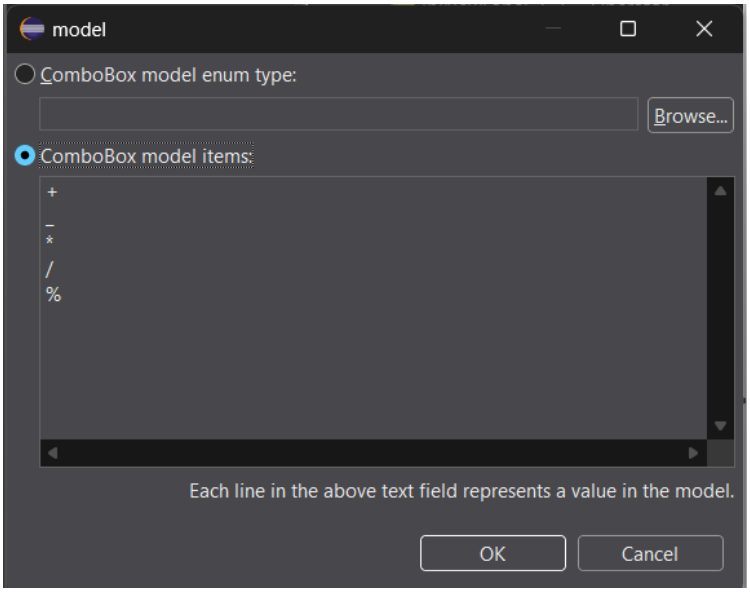
2. Kita akan menggunakan 1 *Layout (Absolute Layout)* dan beberapa komponen pada design aplikasi kita seperti : JLabel., JComboBox, JTextField dan Jbutton. Lalu kita akab memberikan nama atau text yang akan ditampilkan pada tampilan masing masing komponen.



Gambar 2.2 : Komponen Penyusun Design Aplikasi Operator Aritmatika.

Pada gambar 2.2, kita dapat melihat jumlah penggunaan komponen pada design aplikasi. Kita menggunakan 5 label, 3 textField,, 1 combo box dan 1 button. Untuk absolute layout di beri nama Operator Aritmatika. Untuk Label 1 – 5 diberi nama “OPERATOR ARITMATIKA”, “Bilangan 1”, “Bilangan 2”, “Operator”, “Hasil” berturut-turut. Komponen Button diberi nama “Proses” dan untuk komponen textField sendiri tidak diberikan teks yang akan ditampilkan, karenan text Field akan berisi inputan dari *user* dan juga output hasil operasi aplikasi.

3. Langkah selanjutnya kita akan mengisikan model pada komponen combo box yaitu operator aritmatika (+, -, *, /, %)



Gambar 2.3 : Model Item untuk Combo box.

Items model yang kita inputkan pada Combo Box akan memungkinkan pengguna untuk memilih operator yang akan digunakan untuk pilihan operasi. Pada program ini sesuai Namanya, ada operasi penambahan, pengurangan, perkalian, pembagian dan sisa hasil bagi.

4. Lalu, kita akan memberikan variabel pada komponen textField dan Combo Box.
   a. Text Field 1 : txtBil1
   b. Text Field 2 : txtBil2
   c. Text Field 3 : txtHasil
   d. Combo Box : cbOperator

5. Setelah selesai dengan design aplikasi, kita akan beralih pada logic program yang akan kita ubah pada *source code* program kita. Kita bisa klik 2 kali pada komponen Button. Dan akan keluar tampilan seperti ini

```
JButton btnNewButton = new JButton("Proses");
btnNewButton.setFont(new Font("Times New Roman", Font.PLAIN, 12));
btnNewButton.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

```

Gambar 2.4 : Menangani *event* pada tombol Jbutton.

Pada bagian inilah kita akan menulis kode program yang ingin dijalankan ketika tombol “Proses” ini ditekan oleh pengguna.

6. Mendeklarasikan hasil dengan tipe data integer untuk menyimpan hasil operasi.

```
JButton btnNewButton = new JButton("Proses");
btnNewButton.setFont(new Font("Times New Roman", Font.PLAIN, 12))
btnNewButton.addActionListener(new ActionListener() {
    int hasil;
    public void actionPerformed(ActionEvent e) {
```

Gambar 2.5 : Deklarasi Variabel hasil .

7. Menulis logika program pada metode yang akan dijalankan saat tombol "Proses" ditekan.

```
147                 int a = Integer.parseInt(txtBil1.getText());
148                 int b = Integer.parseInt(txtBil2.getText());
149                 int c = cbOperator.getSelectedIndex();
150
151                 if (c==0) {hasil=a+b;}
152                 if (c==1) {hasil=a-b;}
153                 if (c==2) {hasil=a*b;}
154                 if (c==3) {hasil=a/b;}
155                 if (c==4) {hasil=a%b;}
156
157                 txtHasil.setText(String.valueOf(hasil));
```

Gambar 2.6 : Kode Program pada *method actionPerformed.*

Terlihat dari gambar 2.5, pada *line* ke-147 dan ke-148 itu merupakan deklarasi variabel a dan b sebagai integer yang dimanipulasi datanya yang awalnya teks (karena a dan b diambil dari txtBil1 dan txtBil2 yang merupakan inputan dari *user* dan inputan tsb dibaca sebagai teks atau String) menjadi tipe integer agar bisa dilakukan operasi aritmatika oleh aplikasi. Pada *line* ke-149 dapat terlihat *line* tersebut adalah mendeklarasikan variabel c sebagai *indeks* pada tiap *items* yang ada di Combo box sesuai dengan urutannya. *Line* ke- 151 hingga 155 itu merupakan percabangan (*multiple if*) yang akan membaca pilihan operator dari *user* dan mencocokkannya dengan indeks yang ada dan dilakukannya sesuai indeks. Misalnya ketika *user* memilih operator pengurangan (-), pengurangan memiliki indeks ==1, sehingga nilai variabel c==1 dan ketika kondisi tersebut terpenuhi operasi pada nilai inputan (a&b) akan dilakukan sesuai dengan percabangan *multiple if* yaitu pengurangan lalu disimpan pada variabel hasil. Dan untuk *line* ke-157 merupakan kode program yang memungkinkan txtHasil akan menghasilkan hasil dari operasi yang telah dilakukan pada struktur percabangan sebelumnya.

8. Membuat output untuk kemungkinan Error dan adanya kesalahan. Agar program ini efisien, kita memerlukan pesan yang akan muncul ketika pengguna memasukkan data yang tidak valid atau tidak memasukkan data sama sekali. Kita akan membuat dua fungsi *private* dalam kelas program kita untuk menampilkan pesan bawaan Swing dengan tampilan khusus yaitu peringatan dan kesalahan.

```
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); }
```

Gambar 2.7 : Fungsi untuk menampilkan Kesalahan dan Peringatan.

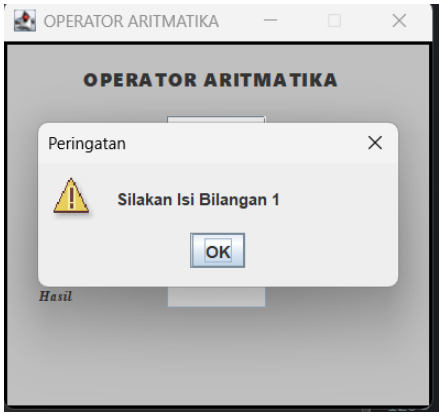
Pertama kita akan memanggil fungsi *pesanPeringatan* dengan percabangan seperti gambar di bawah ini :

```
if(txtBil1.getText().trim().isEmpty()) {
    pesanPeringatan ("Silakan Isi Bilangan 1");
} else if (txtBil2.getText().trim().isEmpty()) {
    pesanPeringatan ("Silakan Isi Bilangan 2");
}else if (txtBil2.getText().trim().startsWith("0")) {
    pesanPeringatan ("Bilangan 2 tidak boleh angka 0");
```

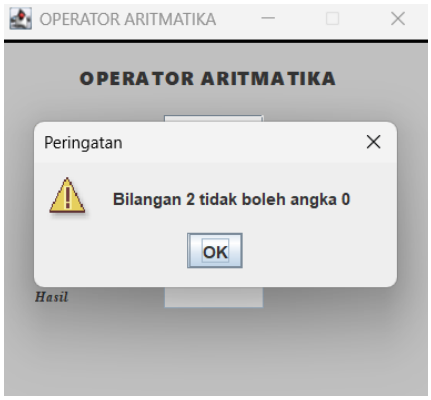
Gambar 2.8 : Pemanggilan Fungsi *pesanPeringatan*..

Pada percabangan *if* pertama fungsi *pesanPeringatan* diminta untuk menampilkan pesan (“Silakan Isi Bilangan 1”) ketika txtBil1 terdeteksi kosong. Begitu pula dengan percabangan *else if* yang pertama akan memanggil fungsi *pesanPeringatan* untuk menampilkan pesan (“Silakan Isi Bilangan 2”) saat txtBil2 terdekteksi kosong. Dan untuk percabangan *else if* yang kedua, fungsi *pesanPeringatan* akan menampilkan pesan (“Bilangan 2 tidak boleh angka 0”) saat txtBil2 dimulai dengan angka 0.

Adapun contoh output dari tampilan pemanggilan fungsi *pesanPeringatan* adalah sebagai berikut :



Gambar 2.9 : Output Program ketika tidak ada inputan untuk Bilangan 1.



Gambar 2.10 : Ouput Program ketika input Bilangan 2 adalah 0.

Lalu setelah fungsi *pesanPeringatan*, selanjutnya kita akan memanggil fungsi ke-2 yaitu fungsi *pesanError*. Fungsi tersebut dipanggil pada logika program saat *user* memasukkan input tidak berupa angka. Kode program pemanggilan fungsi ini menggunakan konsep *try and catch*.

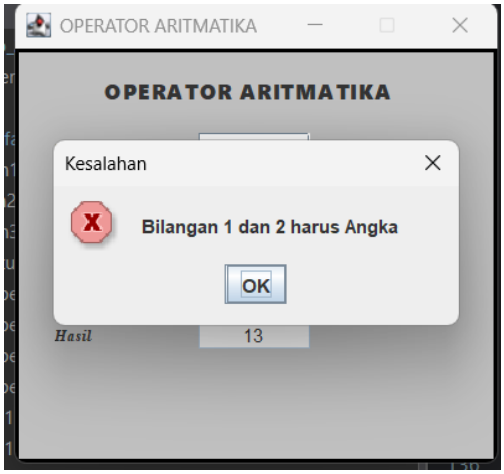
```
} else {
    try {
        int a = Integer.parseInt(txtBil1.getText());
        int b = Integer.parseInt(txtBil2.getText());
        int c = cbOperator.getSelectedIndex();

        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;}

        txtHasil.setText(String.valueOf(hasil));
    } catch (NumberFormatException ex) {
        pesanError("Bilangan 1 dan 2 harus Angka");
    }
}
```

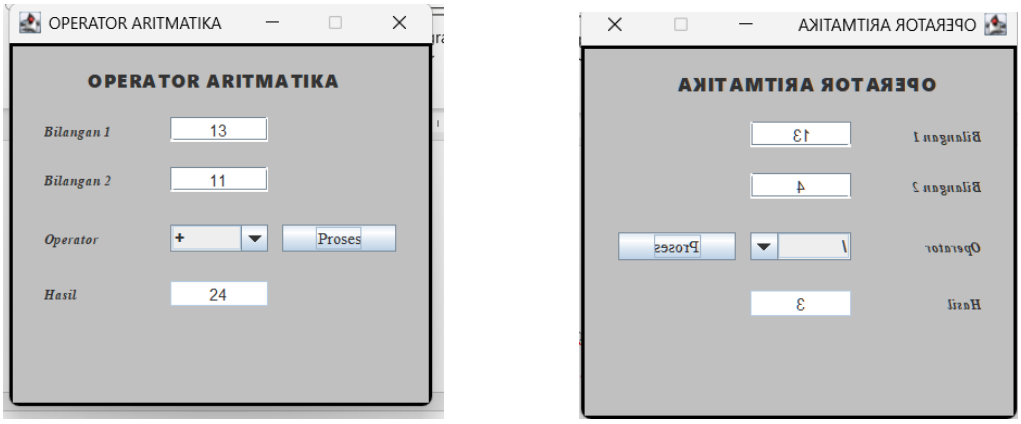
Gambar 2.11 : Kode Program Pemanggilan fungsi *pesanError.*

Potongan program ini adalah lanjutan dari percabangan untuk pemanggilan fungsi sebelumnya. Dan untuk output ketika *user* menginputkan inputan bukan berupa angka akan tertera sebagai berikut :



Gambar 2.12 : Output pemanggilan fungsi *pesanError.*

Setelah menyelesaikan logika program, maka selesailah tahap tahap dari pembuatan aplikasi GUI yang kali ini adalah aplikasi Operator Aritmatika. Berikut Output valid dari aplikasi tersebut:



Gambar 2.12 & 2.13 : Output Aplikasi Operator Aritmatika (Valid).

BAB III
KESIMPULAN

Dari praktikum yang telah dilaksanakan, kita dapat menyimpulkan bahwa praktikum pembuatan aplikasi GUI sederhana ini sudah membantu menguasai pemahaman akan komponen komponen dasar GUI *Swing* seperti JFrame, Jbutton, JtextField, dll. Mekanisme dari *event-driven programming* melalui *listener* seperti *method ActionListener()* memberikan kondisi nyata bagaimana suatu komponen tombol bisa menciptakan interaksi antar *user* dan *program* Dalam praktikum ini juga memperkuat pmahaman akan pengimplementasian *method* seperti *pesanPeringatan* dan *pesanError* dengan tujuan keefektifan dan keefisienan program.

Melalui praktikum ini juga kita dapat menyadari bahwa *plugin* WindowBuilder sangat mempermudah dalam proses pengembangan aplikasi antarmuka secara visual. Dengan konsep *drag-and-drop*, *plugin* WindowBuilder mempercepat transisi dari rancangan ke implementasi kode, WindowBuilder juga masih tetap menghasilkan kode Java murni sehingga pemahaman struktur program dan logika program masih aktif digunakan.

Praktikum ini tentunya diharapkan bisa menjadi fondasi penting untuk pengembangan aplikasi GUI lebih lanjut, sekaligus melatih skil *problem solving* sesuai dengan alur kerja program.

DAFTAR PUSTAKA

[1] *WindowBuilder: A powerful, easy-to-use, bi-directional Java GUI designer*, Eclipse Foundation, 2025. [Daring].Tersedia: https://eclipse.dev/windowbuilder/ . [Diakses:20 November 2025].

[2] Revou, "GUI (Graphical User Interface)," *Revou.co*, 2023. [Daring]. Tersedia: https://www.revou.co/id/kosakata/gui . [Diakses: 20 November 2025].

[3] Eclipse Foundation, *WindowBuilder User Guide*, Eclipse Help Center, 2025. [Daring]. Tersedia: https://help.eclipse.org/latest/index.jsp?topic=%2Forg.eclipse.wb.doc.user%2Fhtml%2Findex.html . [Diakses: 20 November 2025].

[4] Oracle Corporation, *Running Java GUIs*, dalam *Oracle® Developer Studio 8.0 Documentation Library*, 2014. [Daring]. Tersedia: https://docs.oracle.com/cd/E50453_01/doc.80/e50452/run_java_guis.htm . [Diakses: 20 November 2025].

[5] A. Fauzi, "Event Handling GUI pada Java," *7 Seasons*, 30 Oktober 2010. [Daring]. Tersedia: https://7seasons.wordpress.com/2010/10/30/event-handling-gui-pada-java/ . [Diakses: 20 November 2025].

[6] A. D. Kusuma, *Materi Praktikum Pemrograman Berorientasi Objek: GUI dan Event Handling*, Bandung: Universitas Komputer Indonesia (UNIKOM), 2020, hlm. 3–7. [Daring]. Tersedia: https://repository.unikom.ac.id/36213/1/PERTEMUAN%20GUI%20EVENT%20HANDLING.pdf . [Diakses: 20 November 2025].

[7] F. Arifin, "Mengenal JFrame pada Java Swing," *Petani Kode*, 2020. [Daring]. Tersedia: https://www.petanikode.com/java-swing-jframe/ . [Diakses: 20 November 2025].