LAPORAN PRAKTIKUM

PEMROGRAMAN ALGORITMA DAN PEMROGRAMAN

“PROGRAM GUI 2: APLIKASI KALKULATOR”

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-9 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.

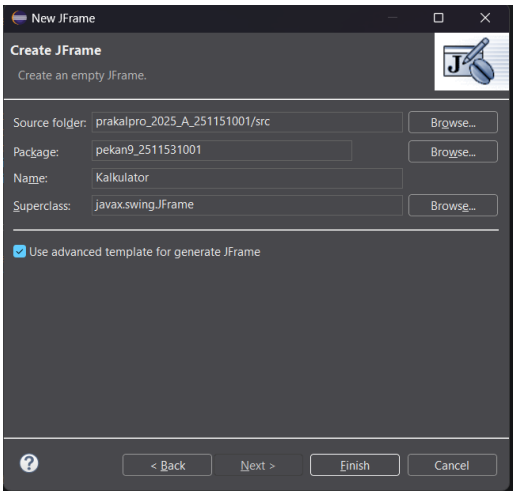
Kalkulator adalah alat bantu manusia yang sudah banyak berkembang dari masa ke masa, dan pengembangan ini bisa dijadikan sebagai ajang pembelajaran dalam pembangunan aplikasi GUI dengan menggunakan bahasa pemrograman Java dengan *plugin* WindowBuilder. Dalam pembangunan aplikasi kalkulator, kita memerlukan pemahaman tentang bagaimana sistem akan merekam dan memperbarui data yang kita masukkan. Hal ini berhubungan karena kita akan membangun aplikasi kalkulator yang mampu mengoperasikan dua bilangan yang berbeda. Dalam kode program ini, kita juga membutuhkan manipulasi tipe string kepada tipe data integer. Hal ini dikarenakan kita akan menampilkan hasil kepada pengguna yaitu berupa teks, sementara kita melakukan interaksi dari data yang diinputkan dengan rumus integer. Dengan demikian, pembangunan aplikasi kalkulator akanmenguji pemahaman akan struktur GUI saja, namun juga alur kendali dan manajemen memori program.

2.2 Program Aplikasi Kalkulator

Sebelumnya kita sudah melakukan penulisan kode program untuk membuat aplikasi operator aritmatika, dan pada praktikum kali ini kita membuat program aplikasi kalkulator dengan tampilan GUI yang berbeda dengan aplikasi yang sudah dibangun sebelumnya. Pada aplikasi ini kita masih tetap melakukan operasi aritmatika sama halnya dengan kalkulator fisik biasa. Namun, pada aplikasi ini kita hanya bisa melakukan dengan dua bilangan saja. Jika *user* menginput tiga atau lebih bilangan, maka bilangan yang akan dieksekusi adalah dua bilangan terakhir.

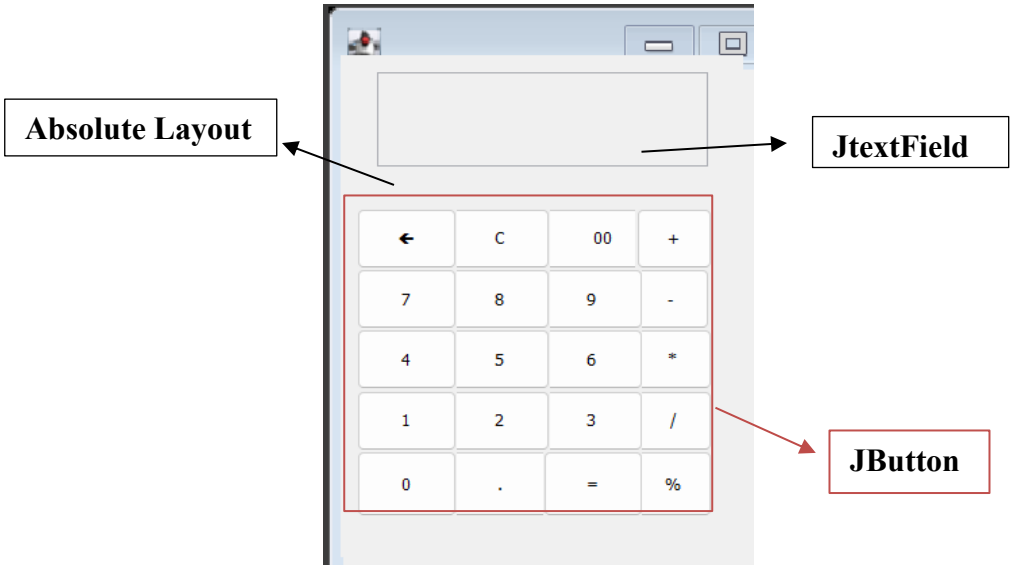
Adapun langkah-langkah praktis untuk membuat GUI dari aplikasi Kalkulator ini adalah sebagai berikut :

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



Gambar 2.1 : Proyek Kalkulator.

2. Pada design aplikasi ini kita akan menggunakan komponen GUI seperti berikut: 1 *absolute layout*, 1 JtextField, dan 20 Jbutton. Adapun gambaran design dari aplikasi kalkulator kita adalah:



Gambar 2.2 : Komponen Penyusun Design Aplikasi Kalkulator.

Untuk mempermudah dalam melihat dan mengetahui masing-masing teks dan variable yang ada pada masing-masing komponen, bisa dilihat pada table 2.1 dibawah ini.

Tabel 2.1 : Tabel Nama Variabel dan Fungsi Komponen Design Aplikasi Kalkulator

No	Teks	Variabel	Fungsi
1	0	angka0	Untuk menyimpan dan menampilkan nilai fisik angka 0
2	1	angka1	Untuk menyimpan dan menampilkan nilai fisik angka 1
3	2	angka2	Untuk menyimpan dan menampilkan nilai fisik angka 2
4	3	angka3	Untuk menyimpan dan menampilkan nilai fisik angka 3
5	4	angka4	Untuk menyimpan dan menampilkan nilai fisik angka 4
6	5	angka5	Untuk menyimpan dan menampilkan nilai fisik angka 5
7	6	angka6	Untuk menyimpan dan menampilkan nilai fisik angka 6
8	7	angka7	Untuk menyimpan dan menampilkan nilai fisik angka 7

9	8	angka8	Untuk menyimpan dan menampilkan nilai fisik angka 8
10	9	angka9	Untuk menyimpan dan menampilkan nilai fisik angka 9
11	0	angka00	Untuk menyimpan dan menampilkan nilai fisik angka 00
12	.	btnDot	Untuk menyimpan dan memberikan nilai pecahan pada inputan *user*
13	=	btnHasil	Untuk menampilkan hasil dari operasi
14	%	btnMod	Untuk menyimpan jenis operator sisa hasil bagi
15	/	btnBagi	Untuk menyimpan jenis operator pembagian
16	*	btnKali	Untuk menyimpan jenis operator perkalian
17	-	btnKurang	Untuk menyimpan jenis operator pengurangan
18	+	btnTambah	Untuk menyimpan jenis operator penambahan
19	C	btnHapus	Untuk menghapus teks yang ada pada komponen JtextField
20	←	btnBackspace	Untuk menghapus satu karakter teks di Jtextfield
21	(none)	txtTampil	Untuk menampilkan angka inputan dari user dan juga hasil atau output dari operasi

3. Setelah menyelesaikan penyusunan design aplikasi, kita akan mulai masuk dalam penulisan kode program. Untuk blok kode program pertama yang akan kita bangun adalah program untuk masing masing tombol angka pada aplikasi.

```
JButton angka0 = new JButton("0");
angka0.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String number = txtTampil.getText()+ angka0.getText();
        txtTampil.setText(number);
    }
});
```

Gambar 2.3 : Kode program untuk menampilkan angka 0.

Sebagai contoh, gambar 2.3 adalah format yang kita gunakan untuk menyimpan angka 0 dan menampilkannya pada layer JtextField. Pada setiap tombol angka kita menggunakan variabel “number” yang bertipe data String dan kita akan mengeset nilai number pada JTextField yang memiliki variabel TxtTampil untuk menampilkan angka tersebut pada

layer txtTampil. Namun, sebelumnya kita harus menyetel variabel "number" dengan method *getText* gabungan dari TxtTampil dan variabel tombol angka. Begitu seterusnya untuk angka-angka 00, dan 1-9.

4. Lalu, kita akan memberikan kode program untuk tombol koma (.). Kode program untuk tombol ini adalah sebagai berikut:

```
JButton btnDot = new JButton(".");
btnDot.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String number = txtTampil.getText()+ btnDot.getText();
        txtTampil.setText(number);
    }
});
```

Gambar 2.4 : Menangani *event* pada tombol btnDot.

Untuk program btnDot juga menggunakan variabel "number" dan untuk menampilkannya juga kita menggunakan *method getText* dan *setText*.

5. Selanjutnya kita akan menangani event pada tombol btnHapus. Kode program untuk pengeksekusian tombol ini adalah sebagai berikut:

```
JButton btnHapus = new JButton("C");
btnHapus.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        txtTampil.setText(null);
    }
});
```

Gambar 2.5 : Menangani *event* pada tombol btnHapus.

Pada bagian ini kita akan menggunakan *method setText* untuk komponen txtHasil untuk menampilkan teks kosong (*null*). Sehingga, angka input ataupun hasilan output yang sebelumnya ditampilkan pada txtTampil akan hilang.

6. Lalu, kita akan menulis program untuk tombol btnBackspace. Kode program adalah sebagai berikut:

```
JButton btnBackspace = new JButton("\uF0E7");
btnBackspace.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String backSpace =null;
        if(txtTampil.getText().length()>0);{
            StringBuilder str = new StringBuilder(txtTampil.getText());
            str.deleteCharAt(txtTampil.getText().length()-1);
            backSpace = str.toString();
            txtTampil.setText(backSpace);
            }
    }
});
```

Gambar 2.5 : Blok program untuk btnBackspace.

Pada blok program ini, kita akan mendeklarasikan variabel “backSpace" dalam tipe data String untuk menyimpan teks setelah penghapusan. Dalam blok ini juga kita menggunakan percabangan *if* untuk memastikan bahwa tidak ada *error* jika layer dalam keadaan kosong dan blok kode berikutnya hanya akan berjalan jika kondisi layar berisi teks. Jika kondisi percabangan terpenuhi, kita menggunakan StringBuilder agar lebih efisien dalam memanipulasi teks lalu menggunakan *method getText* untuk mengambil teks saat ini dari txtTampil. Lalu kita menggunakan *method deleteCharAt()* untuk menghapus karakter di posisi indeks -1 yaitu posisi terakhir. Langkah terakhir untuk blok kode program ini adalah mengkonversi kembali dari StringBuilder menjadi String. Dan menggunakan *method setText* untuk memperbaharui tampilan teks di layar txtTampil dengan teks baru yang sudah dihapus karakter terakhirnya.

7. Step selanjutnya adalah step yang sangat penting yaitu kita akan mendeklarasikan varibel lokal yang dibutuhkan.

```
private static final long serialVersionUID = 1L;
private JPanel contentPane;
private JTextField txtTampil;

    double Bil1;
    double Bil2;
    double hasil;
    String op;
    String jawab;
```

Gambar 2.6 : Deklarasi Variabel.

Variabel “Bil1” akan digunakan sebagai wadah penyimpanan bilangan 1 sedangkan “Bil2” adalah tempat penyimpanan bilangan 2, dan variabel “hasil” akan menyimpan hasil operasi dari bilangan 1 dan bilangan 2, ketiga variabel ini bertipe data double. Sedangkan “op” akan

menyimpan semua jenis operator dan variabel jawab digunakan untuk menyimpan proses atau perumusan berbeda dari masing masing operator hasilkan.

8. Langkah selanjutnya, kita akan menulis logika program pada masing masing operator yang kita punya pada aplikasi kalkulator ini. Kode programnya adalah sebagai berikut

```
JButton btnTambah_2511531001 = new JButton("+");
btnTambah_2511531001.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        Bil1=Double.parseDouble(txtTampil.getText());
        txtTampil.setText("");
        op ="+";
    }
});
```

Gambar 2. 7: Kode Program untuk Operator Penambahan.

Kode program 2.7 adalah kode program dalam menangani *event* untuk tombol operator "+". Blok kode program tersebut berfungsi untuk menyetel operator yang akan disimpan ke dalam variabel "op" tergantung pengguna memilih operator pada aplikasi. Blok kode program untuk operator lainnya tetap sama, hanya butuh penyesuaian pada deklarasi variabel "op" sesuai dengan operator penguranga, perkalian,pembagian, ataupun sisa hasil bagi.

9. Membuat logika pemrograman pada tombol hasil. Tujuan dari kode program ini adalah untuk melaksanakan operasi sesuai dengan operator yang telah dipilih oleh pengguna.

Pada blok program ini kita akan menggunakan percabangan dengan jumlah sebanyak kemungkinan operator yang aplikasi sediakan. Percabangan yang digunakan adalah *if – else If.* Jika salah satu kondisi dalam percabangan terpenuhi maka akan bilangan 1 dan bilangan 2 inputan *user* akan dieksekusi sesuai dengan operator. Hasil dari operasi kedua bilangan akan disimpan pada variabel " hasil".

Setelah blok kode program yang mengandung percabangan telah selesai diesksekusi dan hasil operasi telah disimpan ke dalam variabel "hasil", kita akan menyimpan hasil operasi dengan tipe data String dan dengan format seperti bilangan decimal dengan dua digit yang ikut ditampilkan setelah tanda koma ke dalam variabel "jawab". Dan untuk langkah terakhir, menggunakan *method setText* kita akan menampilkan nilai yang telah disimpan ke dalam variabel "jawab" pada layer txtTampil.

Adapun kode program untuk menangani *event* pada tombol samadengan "=" (btnHasil) adalah sebagai berikut:

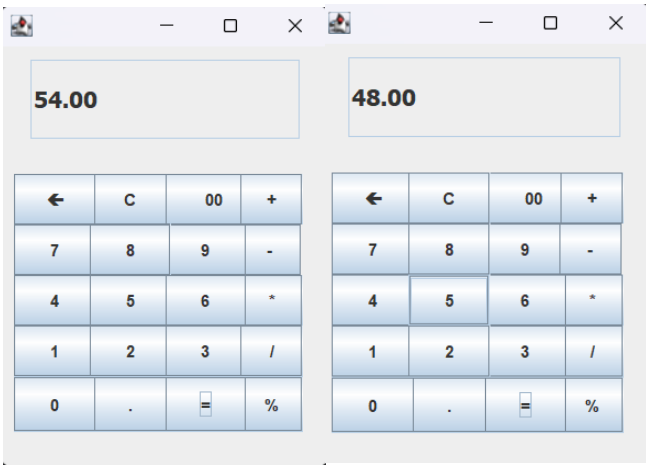
```
JButton btnHasil = new JButton("=");
btnHasil.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        String jawab;
        Bil2=Double.parseDouble(txtTampil.getText());
        if (op=="+") {
            hasil = Bil1 + Bil2;
            jawab = String.format("%.2f",hasil);
            txtTampil.setText(jawab);}
        else if (op=="-") {
            hasil = Bil1 - Bil2;
            jawab = String.format("%.2f",hasil);
            txtTampil.setText(jawab);}
        else if (op=="*") {
            hasil = Bil1 * Bil2;
            jawab = String.format("%.2f",hasil);
            txtTampil.setText(jawab);}
        else if (op=="/") {
            hasil = Bil1 / Bil2;
            jawab = String.format("%.2f",hasil);
            txtTampil.setText(jawab);}
        else if (op=="%") {
            hasil = Bil1 % Bil2;
            jawab = String.format("%.2f",hasil);
            txtTampil.setText(jawab);}

    }
});
```

Gambar 2.8 : Kode program untuk logika operasi.

Setelah menyelesaikan logika program untuk tombol btnHasil, program kita sudah lengkap. Sesuai dengan tujuan aplikasi ini dibangun, dalam pengoperasiannya aplikasi kalkulator ini dapat melakukan operasi penambahan, pengurangan, perkalian, pembagian, dan sisa hasil bagi (modulo) terhadap 2 bilangan. Baik itu kedua bilangannya adalah inputan penggun atau hasil dari operasi sebelumya bisa menjadi bilangan 1 untuk menjalankan operasi selanjutnya. Pengguna juga dapat menghapus seluruh angka di layar atau menghapus digit terakhir dari angka yang ditampilkan, sehingga dapat digunakan sesuai dengan kebutuhan.

Output atau hasil *running* dari aplikasi ini dapat dilihat pada beberapa gambar terlampir. Kita akan mecoba mengalikan angka 9 dan angka 6, lalu mengurangkan hasil dari operasi perkalian tersebut dengan angka 6, dan aplikasi menampilkan hasil sebagai berikut:



Gambar 2.9, Output Hasil Perkalian, 2.10,Output Hasil Pengurangan.

Namun, aplikasi ini masih memiliki kekurangan dan memiliki perbandingan yang jauh dengan kalkulator yang sudah ada pada *device* elektronik. Aplikasi kalkulator ini hanya bisa melakukan operasi pada dua bilangan saja. Namun jika para pengguna memasukkan 3 atau lebih bilangan, aplikasi tidak akan *error* tetapi hanya melakukan operasi pada dua bilangan yang terakhir. Hal ini dikarenakan hanya ada 2 tempat atau wadah yang disediakan oleh aplikasi untuk menyimpan data berupa angka. Ini malah akan berbahaya mengingat hasil yang akan diperoleh jika ketiga atau lebih bilangan itu dioperasikan sesuai dengan keinginan pengguna akan menghasilkan hasil yang berbeda dengan hasil yang ditampilkan oleh aplikasi ini. Namun, aplikasi masih memiliki ruang yang sangat besar untuk dikembangkan kedepannya.

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.

Seterusnya, aplikasi kalkulator sebagai projek dalam mengaplikasikan konsep-konsep pembelajaran telah membantu penulis untuk mengetahui dan memahami cara agar bisa menangani penulisan kode program aplikasi sekaligus membuat tampilan GUI.

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].