

VERSION 1.6

19 Mei 2025



PEMROGRAMAN BERORIENTASI OBJEK

MODUL 6 – JavaFX GUI (Graphical User Interface)

DISUSUN OLEH:

WIRA YUDHA AJI PRATAMA

KEN ARYO BIMANTORO

DIAUDIT OLEH:

Ir. Galih Wasis Wicaksono, S.Kom, M.Cs.

PRESENTED BY: TIM LAB. IT

UNIVERSITAS MUHAMMADIYAH MALANG

TUJUAN

1. Mahasiswa memahami konsep dasar pembuatan antarmuka grafis (GUI) menggunakan **JavaFX**.
2. Mahasiswa mampu mengenal dan menggunakan komponen JavaFX seperti **Stage, Scene, Button, Label, TextField, Pane** dan lainnya.
3. Mahasiswa memahami cara membuat aplikasi yang interaktif dengan **event handling** di JavaFX.
4. Mahasiswa memahami cara mengatur layout dan mengorganisasi komponen GUI menggunakan layout container di JavaFX.

TARGET MODUL

1. Mahasiswa dapat membuat aplikasi GUI sederhana dengan JavaFX.
2. Mahasiswa dapat menambahkan dan mengatur komponen GUI pada scene dan stage.
3. Mahasiswa dapat menangani event, seperti klik tombol, menggunakan **EventHandler** JavaFX.
4. Mahasiswa dapat menggunakan berbagai jenis layout container seperti **VBox, HBox, GridPane** untuk mengatur tampilan aplikasi.

PERSIAPAN

1. Device (Laptop/PC) dengan Java Development Kit (JDK) dan JavaFX SDK terinstall.
2. IDE Java seperti IntelliJ IDEA yang sudah terkonfigurasi dengan JavaFX.
3. Internet dan web browser untuk mencari dokumentasi dan referensi.
4. Pengetahuan dasar Java tentang variabel, tipe data, percabangan, perulangan, dan pemrograman berorientasi objek.

KEYWORDS

JavaFX, Stage, Scene, Button, Label, TextField, Pane, VBox, HBox, GridPane, EventHandler, Scene Builder (opsional)

TABLE OF CONTENTS

TUJUAN.....	1
TARGET MODUL.....	1
PERSIAPAN.....	1
KEYWORDS.....	1
TABLE OF CONTENTS.....	1
BAB I: PENGENALAN GUI DI JAVA.....	3

1.1 APA ITU GUI?	3
1.2 PERBEDAAN CLI DAN GUI	4
1.3 LIBRARY GUI DI JAVA	6
1.4 MENGGUNAKAN GUI (IntelliJ IDEA)	7
BAB II: MEMBUAT GUI DASAR DENGAN JAVAFX	13
2.1 STAGE DAN SCENE	13
2.2 MENAMBAHKAN KOMPONEN (Label, TextField, Button)	15
2.3 LAYOUT DI JAVAFX (VBox dan HBox)	16
2.4 PRAKTIK: FORMULIR SEDERHANA	19
BAB III: EVENT HANDLING (INTERAKSI PENGGUNA)	23
3.1 KONSEP EVENT LISTENER	23
3.2 MENANGANI INPUT DARI TEXT FIELD (JavaFX)	25
3.3 PRAKTIK: KALKULATOR SEDERHANA	26
BAB IV: GUI LEBIH KOMPLEKS DENGAN MODULAR	30
4.1 MENGELOMPOKKAN KOMPONEN DENGAN PANE	30
4.2 KELAS TERPISAH UNTUK GUI (JavaFX)	32
4.3 RADIO BUTTON, CHECKBOX, COMBOBOX	34
4.4 APLIKASI DATA MAHASISWA (JavaFX)	38
4.5 TABEL (JavaFX)	41
BAB V: CODELAB & TUGAS	45
5.1 CODELAB	45
5.2 TUGAS	47
5.3 KRITERIA PENILAIAN	49
5.4 SKALA PENILAIAN	50
5.5 SUMMARY AKHIR MODUL	51

BAB I: PENGENALAN GUI DI JAVA

1.1 APA ITU GUI?

GUI (dibaca "giyu-ai") adalah singkatan dari **Graphical User Interface**. Artinya, GUI adalah **tampilan program yang bisa dilihat dan diklik oleh pengguna**. Misalnya: tombol, kotak untuk nulis teks, menu, dan jendela. Contoh GUI:

NIM	Nama	Jenis Kelamin	HP	Agama	Status

Tanpa GUI, pengguna harus menjalankan program lewat **terminal atau command prompt** (disebut CLI atau Command Line Interface), dan harus ngetik perintah secara manual. Tapi dengan GUI, semuanya lebih gampang. Tinggal **klik tombol, isi data, atau pilih menu** — seperti main aplikasi di HP atau komputer.

Contoh GUI:

- Kalkulator di komputer: kamu tinggal klik angka dan tombol tambah.
- Aplikasi chatting: kamu bisa ngetik pesan di kotak teks dan klik tombol "Kirim".
- Game: kamu bisa klik tombol "Play", atur setting lewat menu, dan sebaga

Kenapa GUI itu penting?

Karena **tidak semua orang suka atau bisa mengetik perintah di terminal**. GUI membuat program jadi **lebih ramah dan mudah digunakan**, terutama oleh orang awam.

GUI di Java

Java menyediakan beberapa alat bantu (disebut "library") untuk membuat GUI. Nanti kita akan pakai salah satunya, yaitu **JavaFX**, untuk membuat program dengan jendela, tombol, kotak teks, dan banyak lagi.

Kesimpulan:

- GUI itu seperti wajah dari program.
- Kalau programnya hanya berupa tulisan di terminal, itu seperti ngobrol lewat kode.
- Tapi kalau programnya punya tombol, jendela, dan menu, itu seperti ngobrol lewat tampilan yang nyaman dan mudah dipahami.

1.2 PERBEDAAN CLI DAN GUI

Dalam dunia pemrograman atau komputer, ada dua cara utama kita berinteraksi dengan sebuah program: **CLI** dan **GUI**.

Apa itu CLI?

CLI adalah singkatan dari **Command Line Interface**. Ini adalah cara lama untuk menggunakan komputer atau program, yaitu **dengan mengetik perintah** lewat terminal (di Windows namanya Command Prompt, di Linux namanya Terminal).

Misalnya kalau kamu mau buka file atau menjalankan program lewat CLI, kamu harus ketik seperti ini:

```
D:\folder_wira\project>bash
(Message from Kali developers)

This is a minimal installation of Kali Linux, you likely
want to install supplementary tools. Learn how:
⇒ https://www.kali.org/docs/troubleshooting/common-minimum-setup/

(Run: "touch ~/.hushlogin" to hide this message)
(LAPTOP-JSF6HF6G)-[/mnt/d/folder_wira/project]
$ cd folder_python/

(LAPTOP-JSF6HF6G)-[/mnt/d/folder_wira/project/folder_python]
$ python3 test.py
Hello World!

(LAPTOP-JSF6HF6G)-[/mnt/d/folder_wira/project/folder_python]
$ |
```

Jadi semua perintah harus diketik satu per satu. Kalau salah ketik, biasanya program nggak jalan. Makanya CLI butuh **ketelitian dan hafalan perintah**.

Apa itu GUI?

GUI adalah **Graphical User Interface**, yaitu cara modern yang lebih nyaman karena **ada tampilannya**. Kamu cukup klik tombol, pilih dari menu, isi kotak teks, dan sebagainya.

Contoh aplikasi GUI:

- Kalkulator
- Microsoft Word
- WhatsApp
- Aplikasi pemutar musik

Kalau kamu buka file di GUI, tinggal klik folder → klik file → selesai. Nggak perlu ketik perintah apa pun.

Perbedaan Sederhana

CLI (Command Line)	GUI (Tampilan Visual)
Harus ngetik perintah	Tinggal klik-klik
Butuh hafalan	Lebih intuitif
Cepat untuk pengguna mahir	Mudah untuk pemula
Contoh: Terminal, CMD	Contoh: WhatsApp, Chrome

Kesimpulan:

- **CLI** cocok untuk pengguna yang sudah terbiasa dan ingin kerja cepat lewat keyboard.
- **GUI** cocok untuk semua orang, karena lebih **ramah pengguna**, terutama untuk orang awam.

Dalam modul ini, kamu akan belajar **membuat GUI**, supaya program buatanmu **bisa dipakai oleh siapa saja** tanpa harus tahu kode atau perintah terminal. Kalau kalian MALAS baca, silahkan nonton video dokumentasi di link berikut:

<https://www.youtube.com/watch?v=FLkOX4Eez6o&list=PL6gx4Cwl9DGBzfXLWLSYVy8EbTdpGbUIG>

https://www.youtube.com/watch?v=_7OM-cMYWbQ&list=PLZPZq0r_RZOM-8vJA3NQFZB7JroDcMwev

<https://www.youtube.com/watch?v=YVXu7tmmQ0M&list=PLlGZc17KPrVAKj3Tl1im5HN8Lh5nYTXyB>

https://www.youtube.com/watch?v=WDaXpDtYk3E&list=PLrzWQu7AjpI26jZvP8JhEJgFPFEj_fojO

https://www.youtube.com/watch?v=9YrmON6nlEw&list=PLS1QulWo1RlaUGP446_pWLgTZPiFizEMq

Tinggal pilih mau marathon yang mana ;)

1.3 LIBRARY GUI DI JAVA

Untuk membuat tampilan antarmuka (GUI) di Java, kita butuh **library**, yaitu semacam "kotak alat" yang sudah disiapkan Java agar kita tidak perlu membuat semuanya dari nol.

Bayangkan kamu mau bikin aplikasi dengan jendela, tombol, dan kotak isian. Kamu tidak perlu menggambar tombol sendiri, cukup **pakai library** yang sudah disediakan Java.

Java punya beberapa library GUI. Yang paling dikenal ada tiga:

AWT (Abstract Window Toolkit)

AWT adalah **library GUI paling dasar di Java**. Isinya sudah ada sejak Java pertama kali dibuat. Tapi tampilannya **kaku dan jadul** (tua), seperti aplikasi zaman majapahit.

- Kelebihan: ringan, sederhana.
- Kekurangan: tampilannya kuno dan kadang tidak sama di tiap sistem operasi.

Contoh: tombol AWT bisa terlihat beda antara Windows dan Linux.

Swing

Swing adalah versi yang lebih **modern dari AWT**. Library ini **lebih lengkap**, tampilannya **lebih menarik**, dan bisa bekerja sama di semua sistem operasi.

- Kelebihan: tampilan konsisten, banyak komponen siap pakai (seperti tombol, label, text field, panel).
- Kekurangan: tampilannya tidak terlalu modern, tapi tetap sangat berguna.

Swing mempunyai kelebihan yaitu:

- Cocok untuk pemula.
- Banyak tutorial dan dokumentasi.
- Tidak perlu alat tambahan.

JavaFX

JavaFX adalah **generasi terbaru** untuk membuat GUI di Java. Tampilannya lebih keren, bisa pakai animasi, efek, grafik, bahkan video.

- Kelebihan: tampilan modern, bisa dipakai untuk aplikasi keren (mirip aplikasi Android).
- Kekurangan: lebih kompleks, butuh alat tambahan seperti Scene Builder kalau mau lebih gampang.

JavaFX cocok untuk kamu yang ingin bikin aplikasi dengan desain modern, kita akan fokus menggunakan **JavaFX** untuk pembelajaran di modul ini.

Kesimpulan

- AWT: dasar, jadul.
- Swing: standar, sering dipakai.
- JavaFX: modern, cocok untuk aplikasi canggih.

1.4 MENGGUNAKAN GUI (IntelliJ IDEA)

IntelliJ IDEA adalah salah satu IDE populer untuk programmer Java. IntelliJ **tidak punya fitur drag & drop** untuk GUI seperti NetBeans, jadi kamu harus **menulis kode GUI secara manual**. Ini justru bagus buat belajar, karena kamu jadi paham betul bagaimana cara kerja komponen GUI di Java.

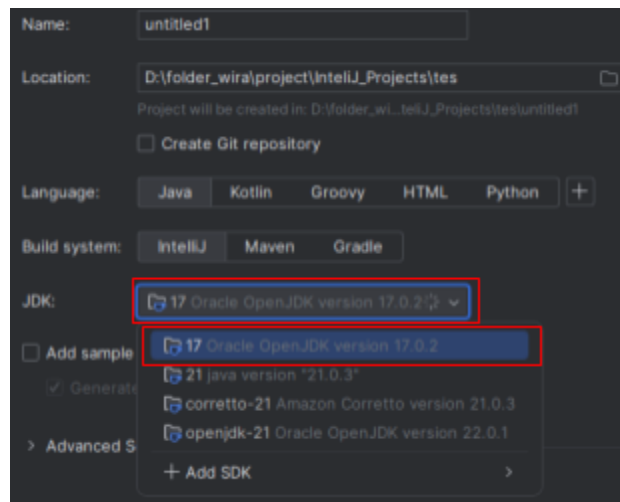
Kenapa Pakai IntelliJ untuk GUI?

- IntelliJ sangat cepat dan cerdas dalam bantu ngecek kode, menemukan kesalahan, dan menyarankan perbaikan.
- Cocok untuk programmer yang ingin belajar kode GUI dari dasar tanpa bantuan “alat otomatis”.
- Banyak developer profesional menggunakan IntelliJ untuk proyek besar.

Cara Membuat GUI di IntelliJ secara Manual

1. Buat Project JavaFX Baru di IntelliJ:

- Pilih **New Project > Java**.
- Pastikan JDK yang dipakai sudah mendukung JavaFX (biasanya JDK 17+).

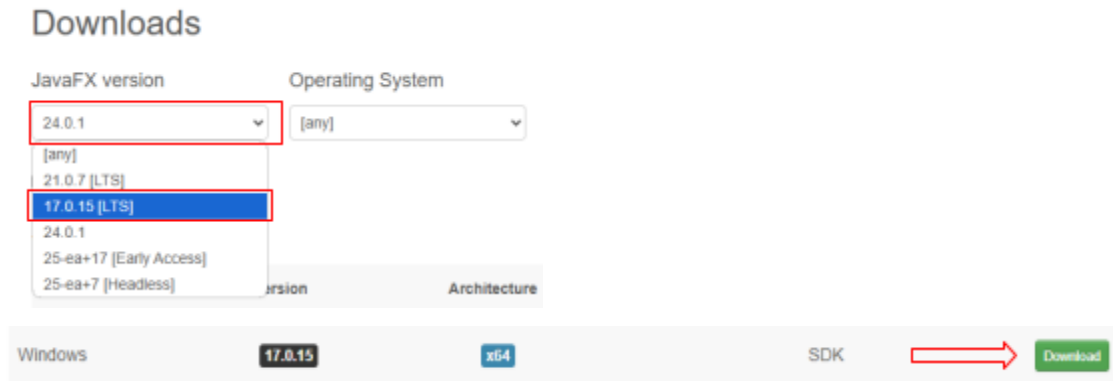


- Kalau belum ada library JavaFX, kamu bisa tambahkan modul JavaFX secara manual atau pakai SDK JavaFX dari website resmi.

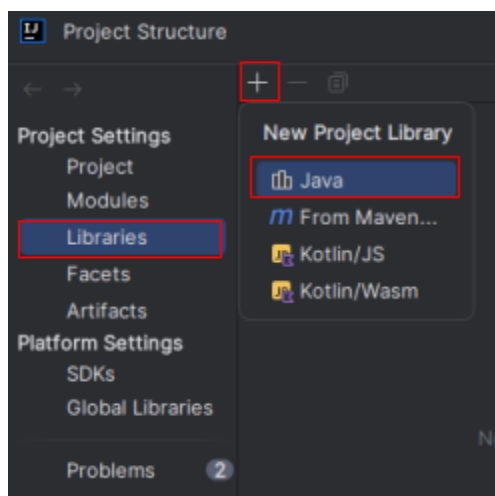
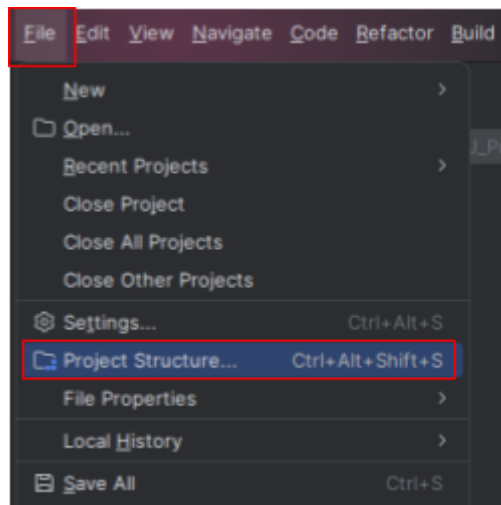
2. Menambahkan Library JavaFX

Kalau belum ada, kamu perlu menambahkan library JavaFX ke project:

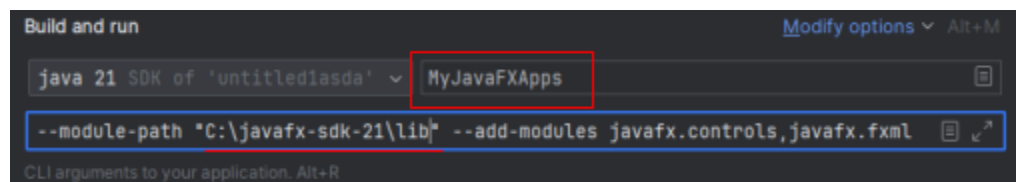
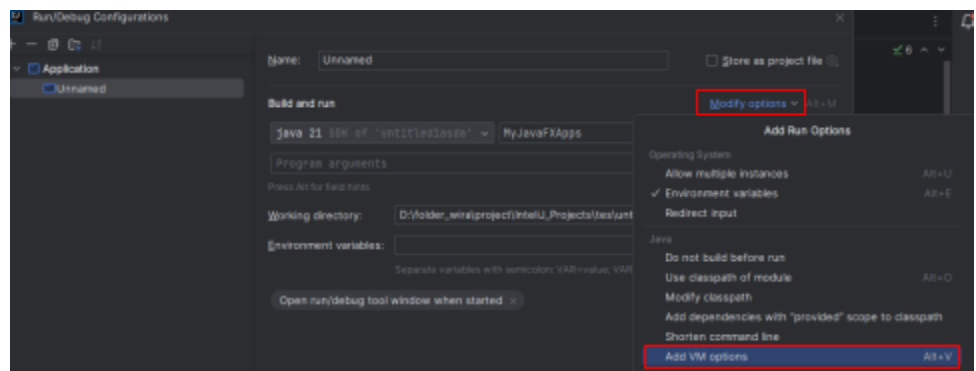
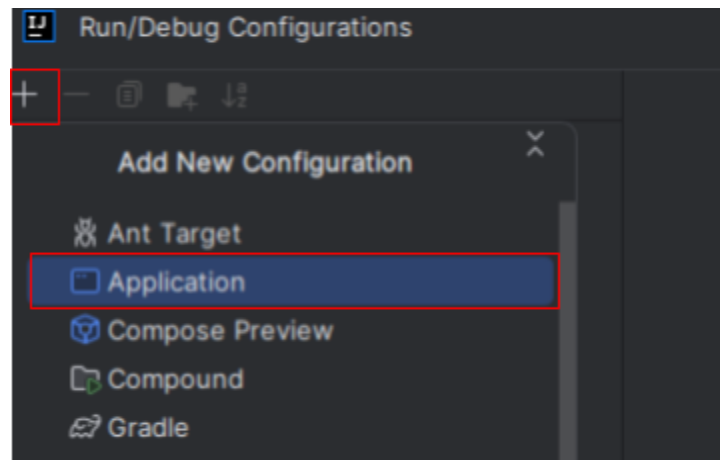
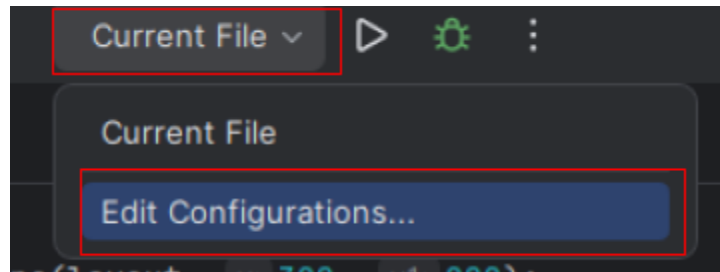
- Download JavaFX SDK dari <https://gluonhq.com/products/javafx/> (Note: Pakai **JavaFX versi 17** saja untuk menghindari error versi tidak kompatibel)



- Setelah itu silahkan extract file .zip nya.
- Di IntelliJ (setelah buat proyek tadi), buka **Project Structure > Libraries**, lalu tambahkan folder lib dari JavaFX SDK di lokasi tempat kalian ekstrak .zip nya. Kemudian tekan OK terus sampai selesai.



- Setelah itu, Atur VM Options saat menjalankan aplikasi. Ikuti contoh dibawah.



- Di bagian “C:\javafx-sdk-21\lib” ganti dengan lokasi lib dari javafx-17 kalian. Untuk bagian Main Class nya isi dengan class yang ada fungsi “main” nya (termasuk dengan packagenya). Jika sudah klik ok terus sampai selesai.

3. Contoh Kode JavaFX Sederhana

```

import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class MyJavaFXApps extends Application {
    @Override
    public void start(Stage primaryStage) {
        Label label = new Label("Halo, ini GUI JavaFX di IntelliJ!");
        Button button = new Button("Klik Aku");

        button.setOnAction(e -> label.setText("Tombol sudah diklik!"));

        VBox layout = new VBox(10);
        layout.getChildren().addAll(label, button);

        Scene scene = new Scene(layout, 300, 200);

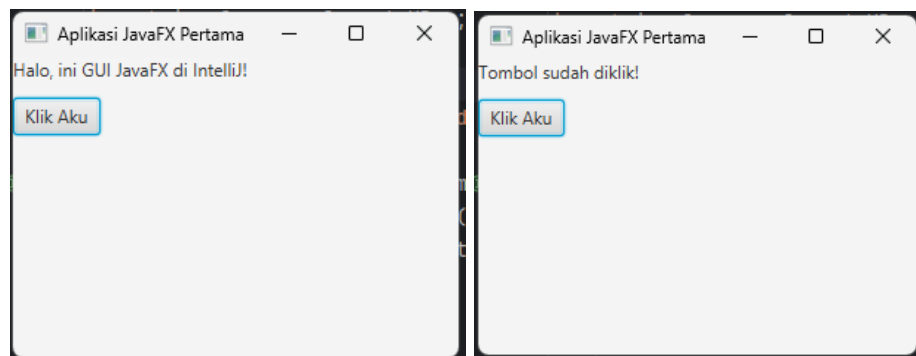
        primaryStage.setTitle("Aplikasi JavaFX Pertama");
        primaryStage.setScene(scene);
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}

```

4. Jalankan Program

- Klik tombol Run di IntelliJ.
- Akan muncul jendela GUI dengan tulisan dan tombol.
- Saat tombol diklik, tulisan pada label berubah.
- Output:



Keuntungan Menggunakan JavaFX

- Tampilan modern dan lebih menarik.
- Lebih mudah buat animasi dan efek visual.
- Layout fleksibel dan mudah diatur.
- Support CSS untuk styling GUI.
- Bisa menggunakan Scene Builder untuk desain GUI drag & drop.

Tips saat menggunakan IntelliJ untuk GUI

- Karena harus menulis kode manual, kamu perlu tahu cara menggunakan **komponen JavaFX** seperti Stage, Scene, Label, Button, dan cara mengatur tata letak (layout) menggunakan **VBox**, **HBox**, **BorderPane**, dan lain-lain.
- Untuk event seperti klik tombol, kamu harus menambahkan kode ActionListener supaya program bisa merespon.

Kesimpulan

- **IntelliJ IDEA** cocok untuk kamu yang ingin belajar dasar GUI Java dengan cara menulis kode langsung.
- Tidak ada drag & drop, jadi kamu harus memahami cara kerja masing-masing komponen.
- Dengan IntelliJ, kamu belajar pemrograman GUI secara mendalam dan fleksibel.

BAB II: MEMBUAT GUI DASAR DENGAN JAVAFX

2.1 STAGE DAN SCENE

Di JavaFX, saat kamu membuat aplikasi dengan tampilan grafis, ada dua hal penting yang harus kamu pahami: **Stage** dan **Scene**.

- **Scene** adalah **isi dari jendela** tersebut, seperti tombol, tulisan, kotak input, dan gambar yang akan kamu lihat dan gunakan.
Jika Stage itu seperti bingkai foto, maka Scene adalah gambar atau foto yang ada di dalam bingkai itu.

```
Scene scene = new Scene(root, 350, 180);
```

Anda bisa membaca dokumentasi tentang Scene lebih lengkap disini:

<https://jenkov.com/tutorials/javafx/stage.html>

<https://docs.oracle.com/javase/8/javafx/api/javafx/stage/Stage.html>

- **Stage** adalah **jendela utama** aplikasi yang muncul di layar komputer kamu. Bayangkan seperti bingkai kosong tempat kamu menaruh semua isi aplikasi. Misalnya, saat kamu membuka aplikasi, yang muncul adalah jendelanya — itulah Stage.

```
primaryStage.setScene(scene);
primaryStage.setTitle("Contoh JavaFX Sederhana");
primaryStage.show();
```

Anda bisa membaca dokumentasi tentang Stage lebih lengkap disini:

<https://jenkov.com/tutorials/javafx/scene.html>

<https://docs.oracle.com/javase/8/javafx/api/javafx/scene/Scene.html>

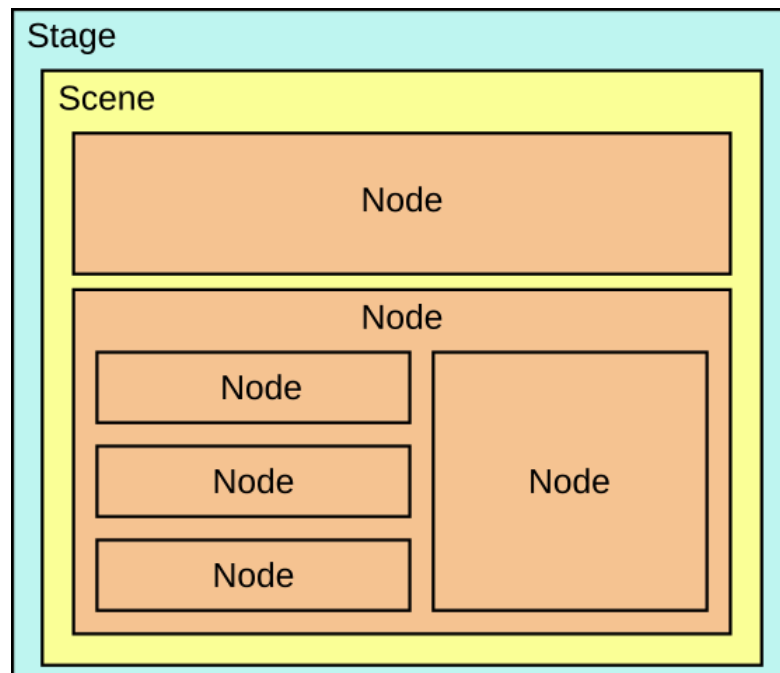
Contoh sederhananya:

Kalau kamu ingin membuat sebuah jendela kosong dalam aplikasi JavaFX, langkah pertama yang perlu kamu lakukan adalah membuat objek Stage dan Scene. Stage bisa kamu anggap sebagai bingkai atau jendela utama dari aplikasi yang akan ditampilkan ke pengguna. Sementara itu, Scene adalah isi dari jendela tersebut, semacam layar kosong tempat kamu nanti akan meletakkan semua komponen antarmuka, seperti tombol, teks, gambar, dan sebagainya.

Setelah kamu membuat Scene, kamu harus menghubungkannya dengan Stage, yaitu dengan cara memasang (set) Scene ke dalam Stage. Ini seperti

memasukkan gambar ke dalam bingkai. Kalau kamu belum memasang Scene, maka jendela yang ditampilkan akan kosong atau bahkan tidak muncul apa-apa.

Langkah terakhir yang sangat penting adalah menampilkan jendela ke layar dengan memanggil fungsi `show()` pada objek Stage. Tanpa langkah ini, jendela tidak akan terlihat meskipun kamu sudah membuat Stage dan Scene. Jadi, urutannya adalah: buat Stage, buat Scene, pasang Scene ke Stage, lalu tampilkan jendelanya dengan `show()`.



Mengapa ini penting?

Karena setiap aplikasi JavaFX pasti membutuhkan setidaknya satu buah Stage, maka bisa dibilang bahwa Stage adalah komponen utama yang berfungsi sebagai jendela tempat semua tampilan aplikasi akan ditampilkan di layar. Tanpa Stage, tidak akan ada wadah untuk menampilkan antarmuka pengguna, sehingga aplikasi tidak bisa muncul di layar.

Namun, Stage sendiri belum cukup. Di dalam Stage, kita juga harus menyisipkan sebuah Scene. Nah, Scene inilah yang berperan sebagai isi atau konten dari jendela tersebut. Bisa kamu bayangkan seperti layar kosong yang nantinya akan diisi berbagai elemen, seperti tombol, teks, gambar, kotak input,

dan sebagainya. Tanpa Scene, Stage hanya akan jadi jendela kosong tanpa isi yang bisa dilihat atau dioperasikan oleh pengguna.

Jadi, hubungan antara keduanya sangat penting: Stage adalah jendelanya, dan Scene adalah tampilan yang mengisi jendela itu. Setiap kali kamu membuat aplikasi JavaFX, kamu perlu membuat keduanya agar pengguna bisa melihat dan berinteraksi dengan antarmuka aplikasi kamu.

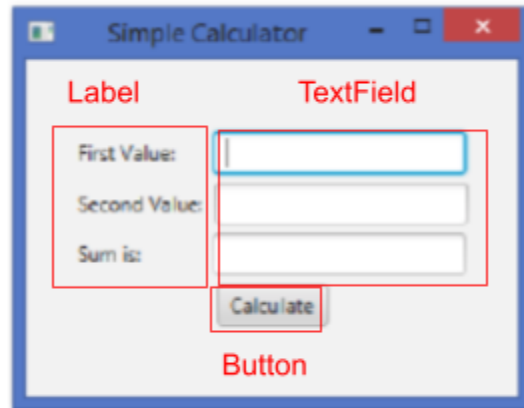
Kalau kamu sudah memahami konsep dasar Stage dan Scene, maka kamu sudah berada di jalur yang tepat untuk mulai membangun antarmuka pengguna (GUI) yang sesungguhnya. Langkah berikutnya adalah mempelajari cara menambahkan berbagai komponen antarmuka (seperti tombol, teks, kolom input, gambar, dan lainnya) ke dalam Scene. Komponen-komponen ini akan memungkinkan aplikasi kamu untuk berinteraksi dengan pengguna, baik melalui klik tombol, pengisian data, atau reaksi terhadap aksi tertentu. Dengan kata lain, kamu akan mulai menghidupkan aplikasi kamu — membuatnya responsif, dinamis, dan benar-benar bisa digunakan oleh orang lain. Tahapan ini adalah pondasi dari pembuatan aplikasi JavaFX yang interaktif dan fungsional.

2.2 MENAMBAHKAN KOMPONEN (Label, TextField, Button)

Setelah kita punya jendela kosong di JavaFX, kita perlu menambahkan **komponen** supaya pengguna bisa berinteraksi dengan program kita.

Komponen itu seperti bagian-bagian di tampilan yang bisa kamu lihat dan gunakan, misalnya:

- **Label:** tulisan yang hanya untuk memberitahu atau memberi informasi, seperti "Nama:".
<https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Label.html>
- **TextField:** kotak tempat pengguna bisa mengetik sesuatu, misalnya mengetik nama atau data lain.
<https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/TextField.html>
- **Button:** tombol yang bisa diklik untuk menjalankan sesuatu, misalnya tombol "Kirim".
<https://docs.oracle.com/javase/8/javafx/api/javafx/scene/control/Button.html>



Cara menambahkan komponen:

1. Buat komponen dengan mudah menggunakan kelas JavaFX yang sudah disediakan. Contohnya:

```
Label label = new Label("Nama:"); // buat tulisan "Nama:"
TextField textField = new TextField(); // buat kotak input teks
Button button = new Button("Kirim"); // buat tombol dengan tulisan "Kirim"
```

2. Tempatkan komponen tersebut ke dalam layout (penyusun tampilan), supaya tampil di jendela. Misalnya kita pakai VBox untuk menata komponen secara vertikal (scroll kebawah untuk materi layout):

```
VBox vbox = new VBox(10); // buat kotak penyusun vertikal, jarak antar komponen 10 piksel
vbox.getChildren().addAll(label, textField, button); // masukkan komponen ke layout
```

Atau:

```
VBox root = new VBox(10, label, textField, button);
```

Kenapa penting?

Dengan komponen ini, program kamu bisa:

- Menampilkan informasi (Label).
- Menerima input dari pengguna (TextField).
- Menerima perintah dari pengguna lewat klik tombol (Button).

Kalau tanpa komponen ini, program cuma tampil kosong dan tidak interaktif. Jadi komponen-komponen ini membuat program jadi **mudah digunakan dan menarik**.

2.3 LAYOUT DI JAVAFX (VBox dan HBox)

Saat kita membuat aplikasi GUI, komponen-komponen seperti tombol, kotak teks, label, dan elemen lainnya tidak bisa diletakkan secara sembarangan di dalam jendela aplikasi. Jika komponen-komponen ini hanya diletakkan asal-asalan, maka tampilan aplikasi akan terlihat berantakan, susah dibaca, dan sulit digunakan oleh pengguna. Oleh karena itu, kita perlu mengatur posisi dan susunan komponen-komponen tersebut dengan rapi dan teratur supaya aplikasi menjadi lebih menarik dan mudah dioperasikan.

Cara untuk mengatur posisi dan susunan komponen di dalam jendela aplikasi ini disebut layout. Layout berfungsi sebagai “kerangka” atau “tata letak” yang menentukan bagaimana dan di mana komponen-komponen GUI ditempatkan. Dengan menggunakan layout yang tepat, kita tidak perlu mengatur posisi komponen secara manual satu per satu, seperti menentukan koordinat x dan y, yang akan sangat merepotkan dan sulit jika aplikasinya kompleks.

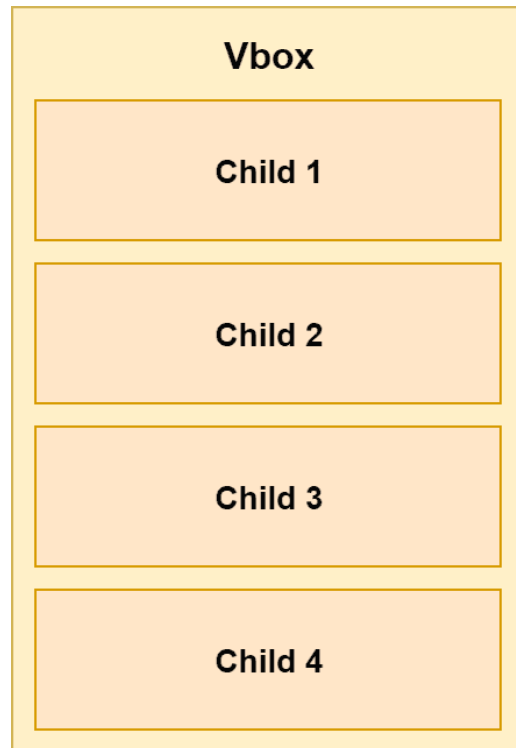
Di JavaFX, ada berbagai jenis layout yang bisa kita gunakan, mulai dari yang sederhana sampai yang lebih kompleks. Namun, untuk pemula dan dalam banyak kasus, dua jenis layout yang paling dasar dan mudah dipahami adalah VBox dan HBox. Kedua layout ini sangat membantu dalam menyusun komponen secara terstruktur dan rapi.

VBox (Vertical Box)

- VBox berfungsi untuk menyusun komponen **secara vertikal**, yaitu dari atas ke bawah.
- Misalnya kamu punya 3 tombol, kalau dimasukkan ke VBox, tombol-tombol itu akan tampil berjejer dari atas ke bawah seperti tangga.
- Kamu juga bisa mengatur jarak antar komponen supaya tidak terlalu rapat, misalnya memberi jarak 10 piksel. Contoh seperti sebelumnya:

```
VBox vbox = new VBox(10); // buat kotak penyusun vertikal, jarak antar komponen 10 piksel
vbox.getChildren().addAll(label, textField, button); // masukkan komponen ke layout
```

Artinya: label, kotak teks, dan tombol akan tersusun vertikal dengan jarak 10 piksel antar mereka.

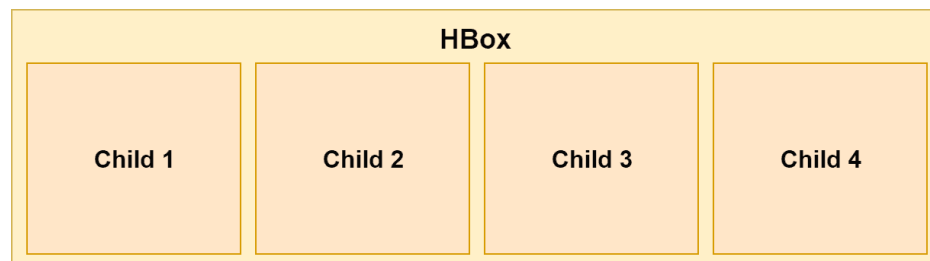


-
- Dokumentasi tentang VBox: <https://jenkov.com/tutorials/javafx/vbox.html>

HBox (Horizontal Box)

- HBox berfungsi untuk menyusun komponen **secara horizontal**, yaitu dari kiri ke kanan.
- Misalnya kamu punya 3 tombol, kalau dimasukkan ke HBox, tombol-tombol itu akan tampil berjajar seperti deretan. Contoh:

```
HBox hbox = new HBox(10); // jarak antar komponen 10 piksel
hbox.getChildren().addAll(button1, button2, button3);
```



-
- Dokumentasi tentang Hbox: <https://jenkov.com/tutorials/javafx/hbox.html>

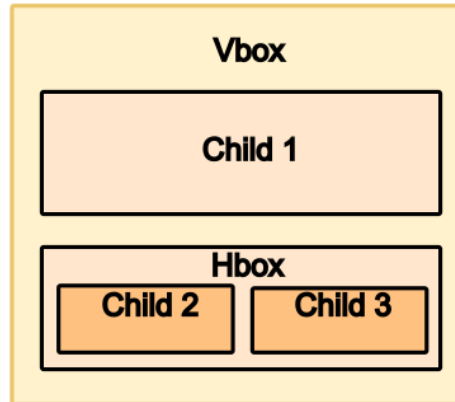
Nested Layout

Kalian juga dapat mengisi layout di dalam layout. Seperti contoh:

```
// Komponen di atas
Label label = new Label("Ini adalah label di dalam VBox");

// HBox di bawah label, isinya dua tombol
Button tombol1 = new Button("Tombol 1");
Button tombol2 = new Button("Tombol 2");
HBox hbox = new HBox(10, tombol1, tombol2); // 10 = spacing antar tombol

// VBox utama
VBox vbox = new VBox(20, label, hbox); // 20 = spacing antar elemen
vbox.setStyle("-fx-padding: 20px;");
```



Kenapa Layout Penting?

- Supaya tampilan aplikasi tidak berantakan.
- Membuat pengguna lebih mudah memahami dan menggunakan aplikasi.
- Memudahkan kita sebagai programmer mengatur tampilan tanpa harus menentukan posisi satu per satu (misal pakai koordinat).

Jadi, dengan VBox dan HBox, kamu tinggal pilih mau susun komponen secara vertikal atau horizontal, lalu tambahkan komponen-komponennya. Gampang dan cepat!

2.4 PRAKTIK: FORMULIR SEDERHANA

Di subbab ini, kamu akan membuat program sederhana menggunakan JavaFX. Program ini akan punya **kotak teks** untuk menulis nama, **tombol** untuk mengirim, dan akan menampilkan **pesan sapaan** seperti "Halo, Wira!" setelah tombol diklik.

Langkah-langkahnya:

1. Buat komponen:

- **Label:** tulisan "Nama:" untuk memberi tahu pengguna harus mengisi apa.
- **TextField:** tempat pengguna mengetik nama.
- **Button:** tombol bertuliskan "Kirim" yang bisa diklik.

2. Atur susunan komponen:

- Gunakan VBox, yaitu layout vertikal agar semua komponen ditumpuk rapi dari atas ke bawah.

3. Buat aksi ketika tombol diklik:

- Ambil teks yang ditulis di TextField.
- Tampilkan pesan pop-up (pakai Alert) yang menyapa pengguna sesuai nama yang ditulis.

Penjelasan Sederhana Kode Programnya:

```
Label label = new Label("Nama");
```

Ini adalah label untuk menampilkan text.

```
TextField textField = new TextField();
```

Ini adalah kotak untuk pengguna mengetik nama.

```
Button button = new Button("Kirim");
```

Ini adalah tombol yang akan diklik pengguna.

```
VBox root = new VBox(10, label, textField, button);
```

Ini adalah VBox untuk menyusun ketiga komponen secara vertikal.

```
Scene scene = new Scene(root, 350, 180);
```

Ini akan memasukan layout vbox sebelumnya (yang namanya 'root') tadi ke dalam scene dengan size 350x180

```
primaryStage.setScene(scene);
primaryStage.setTitle("Contoh JavaFX Sederhana");
primaryStage.show();
```

Ini memasukan scene tadi ke stage kita (primaryStage) pada argumen "start()", kemudian kasih title stagenya, kemudian tampilkan.

```
button.setOnAction(e -> {
    String nama = textField.getText();
    Alert alert = new Alert(Alert.AlertType.INFORMATION);
    alert.setContentText("Halo, " + nama + "!");
    alert.showAndWait();
});
```

Saat tombol diklik:

- Program mengambil teks dari textField (nama yang diketik pengguna).
- Lalu membuat **pop-up** yang menampilkan pesan "Halo, [nama]!"

Keseluruhan kode:

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class MyJavaFXApps extends Application {

    @Override
    public void start(Stage primaryStage) {
        // Buat label baru
        Label label = new Label("Nama");
        // Buat textfield baru
        TextField textField = new TextField();
        // Buat tombol baru
        Button button = new Button("Kirim");

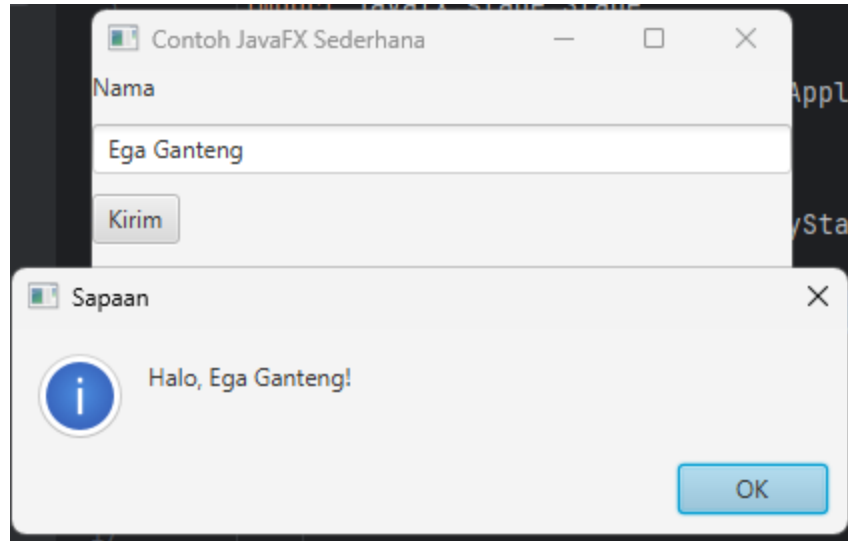
        // Buat EventListener baru
        button.setOnAction(e -> {
            String nama = textField.getText();
            Alert alert = new Alert(Alert.AlertType.INFORMATION);
            alert.setTitle("Sapaan");
            alert.setHeaderText(null);
            alert.setContentText("Halo, " + nama + "!");
            alert.showAndWait();
        });

        // Layout pane
        VBox root = new VBox(10, label, textField, button);
        // Scene yang memuat layout
        Scene scene = new Scene(root, 350, 180);
        // Pasang scene ke stage
        primaryStage.setScene(scene);
        primaryStage.setTitle("Contoh JavaFX Sederhana");
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Hasil Akhirnya:

Ketika kamu jalankan program ini:



- Akan muncul jendela berisi label, kotak teks, dan tombol.
- Kamu bisa mengetik nama kamu, misalnya: **Budi**.
- Klik tombol **Kirim**, maka akan muncul kotak pesan bertuliskan “**Halo, Budi!**”

Dengan latihan ini, kamu sudah mempraktikkan:

- Cara membuat tampilan (GUI).
- Cara menangani aksi pengguna (event handling).
- Cara membuat aplikasi sederhana tapi interaktif.

Kalau kamu sudah paham yang ini, kamu siap lanjut ke membuat aplikasi yang lebih kompleks

BAB III: EVENT HANDLING (INTERAKSI PENGGUNA)

3.1 KONSEP EVENT LISTENER

Bayangkan kamu sedang menggunakan sebuah aplikasi, lalu kamu **klik tombol**, **ngetik di kolom**, atau **memilih menu**. Semua hal yang kamu lakukan itu disebut **event** (kejadian).

Dalam dunia pemrograman GUI, **event** adalah segala sesuatu yang dilakukan pengguna di tampilan aplikasi. Misalnya:

- Klik tombol → itu event.
- Mengetik sesuatu → itu event.
- Menggerakkan mouse → juga event.

Tapi... event **tidak akan berguna** kalau program tidak tahu harus melakukan apa. Maka dari itu, kita butuh yang namanya **listener**.

Apa itu Listener?

Listener artinya "pendengar". Ini adalah bagian dari program yang **mendengarkan** event tertentu. Kalau ada event terjadi (misalnya tombol diklik), listener akan langsung **menjalankan aksi yang sudah ditentukan**.

- **Event:** kejadian di aplikasi (misalnya tombol diklik).
- **Listener:** pendengar yang merespon event (misalnya tampilkan pesan kalau tombol diklik)

Contoh Sederhana (Ibarat Kehidupan Nyata)

- **Tombol Bel Rumah** = Event
- **Kamu yang mendengar bel** = Listener
- **Kamu membuka pintu setelah dengar bel** = Aksi

Jadi, dalam program Java, kamu akan bilang: *"Kalau tombol ini ditekan (event), jalankan kode ini (listener)."*

Contoh di Java:

```
Button tombol = new Button("Klik Saya");
tombol.setOnAction(e -> {
    System.out.println("Tombol ditekan!");
});
```


Penjelasan:

- Button = tombol yang bisa diklik di JavaFX (pengganti JButton di Swing)
- setOnAction(...) = memberi tahu tombol, “kalau kamu diklik, lakukan ini ya” (pengganti addActionListener(...) di Swing)
- System.out.println(...) = aksi yang akan dijalankan saat tombol diklik, misalnya menampilkan teks ke konsol
- e -> { ... } adalah **aksi** yang akan dijalankan saat tombol ditekan.
- System.out.println(...) hanya menampilkan teks ke console.

Penjelasan lambda di Java 8+ (->)

Lambda expression adalah cara menulis fungsi atau blok kode yang bisa dipakai langsung di tempat (anonymous function), tanpa harus membuat method atau kelas baru secara terpisah. Lambda memperpendek kode dan membuatnya lebih mudah dibaca.

Sebelum Java 8, kalau kamu ingin meng-handle event, kamu harus buat kelas **anonymous class** yang cukup panjang. Contoh:

```
btn.setOnAction(new EventHandler<ActionEvent>() {
    @Override
    public void handle(ActionEvent event) {
        System.out.println("Tombol diklik!");
    }
});
```

Namun karena Java 8 mendukung lambda, kita bisa tulis lebih ringkas:

```
btn.setOnAction(e -> System.out.println("Tombol diklik!"));
```

Untuk satu blok kode, atau:

```
btn.setOnAction(e -> {
    System.out.println("Klik");
    System.out.println("Event: " + e);
});
```

Untuk lebih sari 1 blok kode.

```
// Lambda dengan satu parameter tanpa tipe dan tanpa kurung
x -> x * x

// Lambda dengan dua parameter dan tipe eksplisit
(int a, int b) -> a + b

// Lambda dengan dua parameter, beberapa pernyataan dalam body
(a, b) -> {
    int result = a + b;
    System.out.println("Hasil penjumlahan: " + result);
    return result;
}
```

Anda tidak usah pusing gimana cara buatnya. Karena di library JavaFX sudah dibuatin. Anda tinggal pakek saja.

Kenapa Ini Penting?

Tanpa adanya listener, tombol pada sebuah aplikasi hanyalah gambar statis yang tampak di layar. Meskipun kamu bisa melihat tombol tersebut dan mungkin mencoba untuk mengkliknya, sebenarnya tombol itu tidak akan melakukan apa pun karena program tidak tahu bagaimana harus merespon aksi klikmu. Jadi, ketika kamu mengklik tombol tersebut, tidak ada perubahan, tidak ada pesan yang muncul, atau tidak ada fungsi yang berjalan.

Listener-lah yang membuat tombol tersebut menjadi hidup dan interaktif. Listener bertugas untuk “mendengarkan” setiap tindakan yang dilakukan pengguna, seperti klik tombol, dan kemudian menjalankan perintah atau aksi tertentu yang sudah kita tentukan sebelumnya. Dengan adanya listener, saat tombol diklik, program tahu harus berbuat apa, misalnya menampilkan pesan, membuka jendela baru, mengubah data, atau menghitung sesuatu.

Jadi, listener adalah penghubung penting antara apa yang pengguna lakukan dengan bagaimana program merespon. Tanpa listener, tombol hanya akan menjadi bagian visual yang tidak berguna, sementara dengan listener, tombol menjadi alat interaksi yang memungkinkan pengguna berkomunikasi dengan aplikasi dan membuat aplikasi itu terasa hidup dan dinamis.

3.2 MENANGANI INPUT DARI TEXT FIELD (JavaFX)

Sama seperti di Swing, di **JavaFX** kita juga bisa membuat kotak isian (input field) agar pengguna bisa mengetik sesuatu di dalam aplikasi kita. Kotak isian ini memungkinkan pengguna memasukkan data, misalnya nama, alamat, atau angka, lalu kita bisa

mengambil data itu untuk **diproses** lebih lanjut, misalnya menampilkan pesan, melakukan perhitungan, atau menyimpan ke database.

Apa itu TextField di JavaFX?

TextField adalah salah satu **komponen antarmuka (GUI)** di JavaFX yang digunakan untuk **menerima input berupa teks dari pengguna**. Komponen ini biasanya ditampilkan sebagai **kotak kosong** di mana pengguna bisa mengetik sesuatu, seperti nama, alamat email, angka, atau informasi lainnya.

Cara Mengambil Teks dari TextField

Untuk mengambil teks yang diketik di TextField, kita pakai:

```
textField.getText();
```

Berikut dokumentasi nya: <https://jenkov.com/tutorials/javafx/textfield.html>

3.3 PRAKTIK: KALKULATOR SEDERHANA

Di subbab ini, kita akan membuat program GUI yang sangat sederhana yaitu **kalkulator penjumlahan dua angka**.

Apa yang akan dilakukan program ini?

Program akan meminta kamu memasukkan dua angka, lalu ketika kamu menekan tombol “Tambah”, program akan menghitung hasil penjumlahan kedua angka tersebut dan menampilkan hasilnya.

Langkah-langkah membuatnya:

1. **Buat dua kotak isian (TextField)** untuk tempat kamu mengetik angka pertama dan angka kedua.
Misalnya:
 - Kotak isian pertama untuk angka pertama.
 - Kotak isian kedua untuk angka kedua.
2. **Buat tombol “Tambah” (JButton)** yang bisa diklik. Saat tombol ini diklik, program akan membaca angka dari kedua kotak isian tadi.
3. **Ambil angka dari kotak isian**, lalu ubah dari bentuk teks menjadi angka yang bisa dijumlahkan. Karena yang kita ketik di kotak isian itu berupa teks, kita harus mengubahnya dulu ke angka menggunakan fungsi `Integer.parseInt()`.
4. **Hitung hasil penjumlahan** dari dua angka tersebut.

5. **Tampilkan hasilnya** dalam sebuah kotak dialog (message box) supaya pengguna bisa melihat hasilnya.
6. **Tangani kemungkinan kesalahan** jika yang diketik bukan angka, supaya program tidak error. Misalnya, jika kamu mengetik huruf, program akan menampilkan pesan peringatan supaya memasukkan angka yang benar.

Contoh kode sederhananya:

Contoh source code:

```
import javafx.application.Application;
import javafx.geometry.Insets;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class MyJavaFXApps extends Application {

    @Override
    public void start(Stage primaryStage) {
        // Membuat kotak isian untuk angka pertama dan kedua
        TextField angka1 = new TextField();
        angka1.setPromptText("Masukkan angka pertama");

        TextField angka2 = new TextField();
        angka2.setPromptText("Masukkan angka kedua");

        // Membuat tombol "Tambah"
        Button tambah = new Button("Tambah");

        // Label untuk menampilkan hasil
        Label hasilLabel = new Label();
```

```

// Menangani event klik tombol
tambah.setOnAction(e -> {
    try {
        int a = Integer.parseInt(angka1.getText());
        int b = Integer.parseInt(angka2.getText());
        int hasil = a + b;
        hasilLabel.setText("Hasil: " + hasil);
    } catch (NumberFormatException ex) {
        hasilLabel.setText("Masukkan angka yang benar!");
    }
});

// Mengatur layout dengan VBox
VBox root = new VBox(10, angka1, angka2, tambah, hasilLabel);
root.setPadding(new Insets(20));

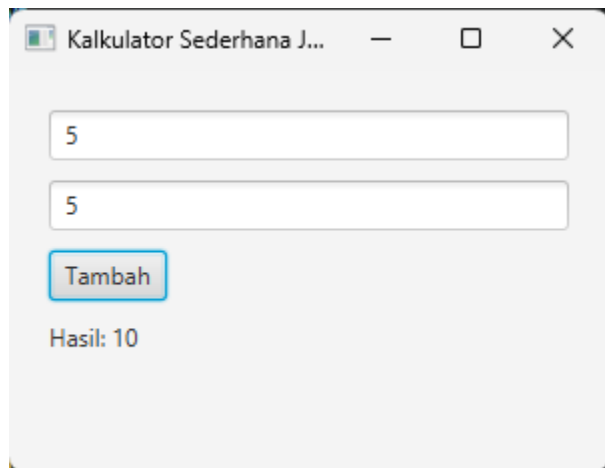
Scene scene = new Scene(root, 300, 200);

primaryStage.setTitle("Kalkulator Sederhana JavaFX");
primaryStage.setScene(scene);
primaryStage.show();
}

public static void main(String[] args) {
    launch(args);
}
}

```

Contoh output:



Penjelasan singkat:

- Kita buat dua tempat input angka dan satu tombol tambah.
- Saat tombol tambah diklik, program mencoba mengubah isi kotak isian menjadi angka.
- Kalau berhasil, hasil penjumlahan ditampilkan.
- Kalau gagal (misal input huruf), program kasih pesan supaya inputnya diperbaiki.

Dengan contoh sederhana ini, kamu sudah belajar cara membuat GUI yang interaktif dan bisa merespon aksi pengguna dengan baik!

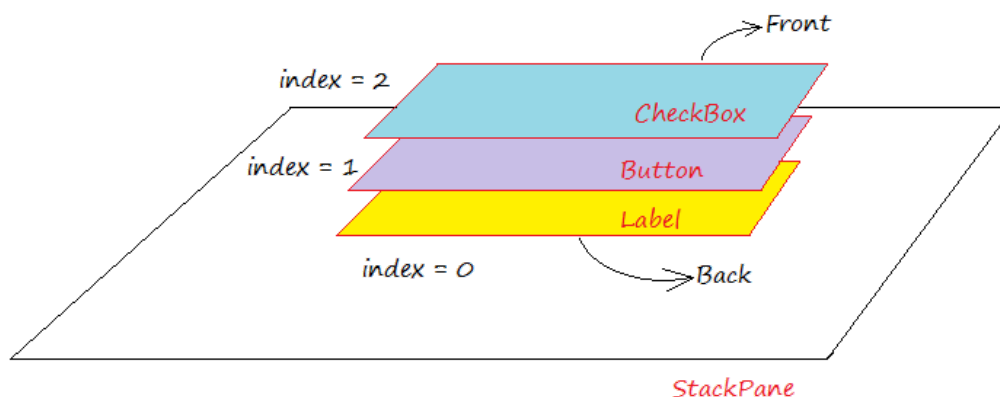
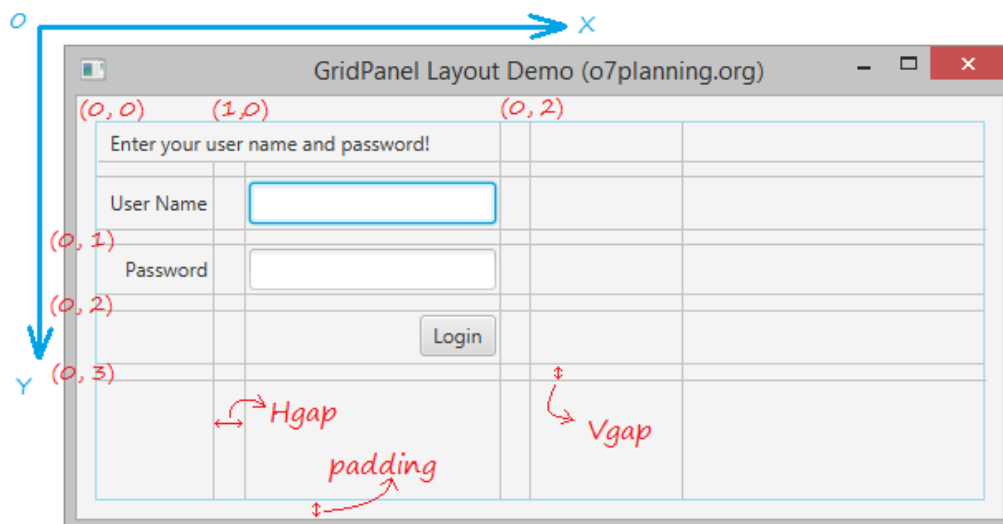
BAB IV: GUI LEBIH KOMPLEKS DENGAN MODULAR

4.1 MENGELOMPOKKAN KOMPONEN DENGAN PANE

Di JavaFX, konsepnya mirip seperti JPanel di Swing, tapi kita menggunakan **Pane** atau jenis container lain untuk mengelompokkan komponen GUI.

Bayangkan kamu punya jendela aplikasi yang isinya banyak tombol, label, dan kolom input. Kalau semua komponen langsung dimasukkan ke jendela utama (Stage dan Scene) tanpa pengelompokan, akan susah mengatur tata letaknya dan terlihat berantakan.

Nah, **Pane** di JavaFX adalah wadah yang bisa kita gunakan untuk mengelompokkan beberapa komponen itu, supaya lebih mudah diatur dan tampilannya rapi.



Beberapa jenis Pane yang sering dipakai:

- **VBox** : Mengatur komponen secara vertikal (atas ke bawah).
- **HBox** : Mengatur komponen secara horizontal (kiri ke kanan).
- **GridPane** : Mengatur komponen dalam bentuk baris dan kolom seperti tabel.
- **StackPane** : Menumpuk komponen di atas satu sama lain.

Dengan membagi komponen ke dalam beberapa Pane, kamu bisa mengelompokkan bagian-bagian GUI, misalnya:

- Satu VBox untuk menaruh label dan text field sebagai form.
- Satu HBox untuk tombol-tombol.
- Lalu semuanya dimasukkan ke dalam root Pane utama.

Contoh sederhana menggunakan VBox dan HBox di JavaFX:

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.HBox;
import javafx.scene.layout.VBox;
import javafx.stage.Stage;

public class MyJavaFXApps extends Application {
    @Override
    public void start(Stage stage) {
        // Membuat label dan text field
        Label label = new Label("Nama:");
        TextField textField = new TextField();
```



```

// Mengelompokkan label dan text field secara vertikal
VBox form = new VBox(10, label, textField);

// Membuat tombol dan menaruhnya secara horizontal
Button btnSimpan = new Button("Simpan");
Button btnBatal = new Button("Batal");
HBox tombolBox = new HBox(10, btnSimpan, btnBatal);

// Mengelompokkan semua ke dalam VBox utama
VBox root = new VBox(20, form, tombolBox);

// Membuat scene dan menampilkannya
Scene scene = new Scene(root, 300, 200);
stage.setScene(scene);
stage.setTitle("Contoh Menggunakan Pane di JavaFX");
stage.show();
}

public static void main(String[] args) {
    launch();
}
}

```

Kesimpulannya:

- Pane di JavaFX itu seperti kotak atau wadah untuk mengelompokkan komponen GUI.
- Memakai Pane membuat tampilan lebih rapi dan mudah diatur.
- Ada berbagai jenis Pane, kamu pilih sesuai kebutuhan pengaturan layout. Silahkan cek dokumentasinya di internet.

4.2 KELAS TERPISAH UNTUK GUI (JavaFX)

Kalau kita membuat aplikasi JavaFX, biasanya semua tampilan (GUI) ditulis di dalam `start(Stage stage)` dalam satu kelas. Tapi kalau aplikasi mulai rumit dan banyak tampilan, akan lebih baik kalau kita **pisahkan tampilan ke dalam kelas khusus**. Tujuannya adalah supaya program tetap rapi, mudah dibaca, dan gampang dikembangkan.

Kenapa Dipisah?

Bayangkan kamu menulis novel, tapi semua bab dan paragraf ditulis tanpa jeda di satu halaman. Pasti bingung membacanya, kan?

Nah, begitu juga dalam program. Kalau semua tampilan dicampur dalam satu kelas, kita akan cepat pusing saat mau mengubah atau menambah sesuatu.

Dengan memisahkan GUI ke dalam kelas khusus, kita bisa:

- Fokus mengatur tampilan di satu tempat.
- Pisahkan logika program dan tampilan supaya tidak bercampur.
- Kerja tim jadi lebih mudah, karena tampilan dan logika bisa dikerjakan secara terpisah.

Contoh Program JavaFX (Dipisah)

Kelas GUI: MyForm.java

```
import javafx.scene.Scene;
import javafx.scene.control.Button;
import javafx.scene.control.Label;
import javafx.scene.control.TextField;
import javafx.scene.layout.VBox;
import javafx.geometry.Pos;
import javafx.scene.layout.HBox;

public class MyForm {
    public Scene getScene() {
        Label label = new Label("Nama:");
        TextField textField = new TextField();
        Button button = new Button("Kirim");

        // Saat tombol diklik, tampilkan nama
        button.setOnAction(e -> {
            System.out.println("Halo, " + textField.getText());
        });

        HBox inputBox = new HBox(10, label, textField);
        VBox layout = new VBox(20, inputBox, button);
        layout.setAlignment(Pos.CENTER);

        return new Scene(layout, 300, 200);
    }
}
```

Kelas Utama: MyJavaFXApps.java

```
import javafx.application.Application;
import javafx.stage.Stage;

public class MyJavaFXApps extends Application {
    @Override
    public void start(Stage primaryStage) {
        MyForm myForm = new MyForm();
        primaryStage.setTitle("Form JavaFX");
        primaryStage.setScene(myForm.getScene());
        primaryStage.show();
    }

    public static void main(String[] args) {
        launch(args);
    }
}
```

Apa yang Terjadi di Sini?

- MyForm adalah **kelas terpisah** yang hanya mengatur tampilan GUI.
- Di dalamnya, kita buat Label, TextField, dan Button.
- Semua disusun dengan VBox dan HBox supaya tampilannya rapi.
- Kelas MainApp hanya fokus menampilkan form ke layar.

Manfaatnya Apa?

- **Lebih rapi:** Kode tampilan tidak bercampur dengan kode utama.
- **Mudah dikembangkan:** Mau ganti tampilan? Tinggal ubah di MyForm.

4.3 RADIO BUTTON, CHECKBOX, COMBOBOX

Saat membuat tampilan aplikasi yang interaktif, kita sering butuh komponen seperti pilihan jenis kelamin, daftar jurusan, atau persetujuan dari pengguna. Di JavaFX, kamu bisa membuat komponen-komponen itu dengan lebih modern dan fleksibel.

RadioButton (Pilihan Satu dari Beberapa)

RadioButton digunakan jika kamu ingin pengguna **memilih salah satu** dari beberapa opsi. Misalnya: Jenis kelamin.

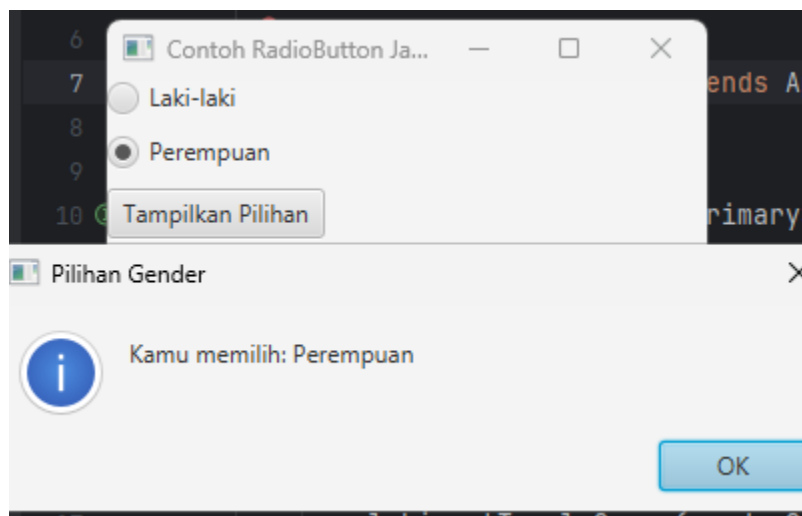
```
// Buat RadioButton
RadioButton laki = new RadioButton("Laki-laki");
RadioButton perempuan = new RadioButton("Perempuan");

// Kelompokkan agar hanya satu bisa dipilih
ToggleGroup genderGroup = new ToggleGroup();
laki.setToggleGroup(genderGroup);
perempuan.setToggleGroup(genderGroup);

// Buat tombol untuk menampilkan pilihan
Button tombol = new Button("Tampilkan Pilihan");

// Aksi saat tombol ditekan
tombol.setOnAction(e -> {
    RadioButton selected = (RadioButton) genderGroup.getSelectedToggle();
    if (selected != null) {
        String pilihan = selected.getText();
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Pilihan Gender");
        alert.setHeaderText(null);
        alert.setContentText("Kamu memilih: " + pilihan);
        alert.showAndWait();
    } else {
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle("Peringatan");
        alert.setHeaderText(null);
        alert.setContentText("Silakan pilih jenis kelamin terlebih dahulu.");
        alert.showAndWait();
    }
});
```

Kalau kamu pilih "Laki-laki", maka "Perempuan" akan otomatis tidak terpilih.



<https://jenkov.com/tutorials/javafx/radiobutton.html>

CheckBox (Bisa Pilih Banyak Opsi)

CheckBox digunakan kalau pengguna **boleh memilih lebih dari satu**.

Contohnya: minat atau langganan.

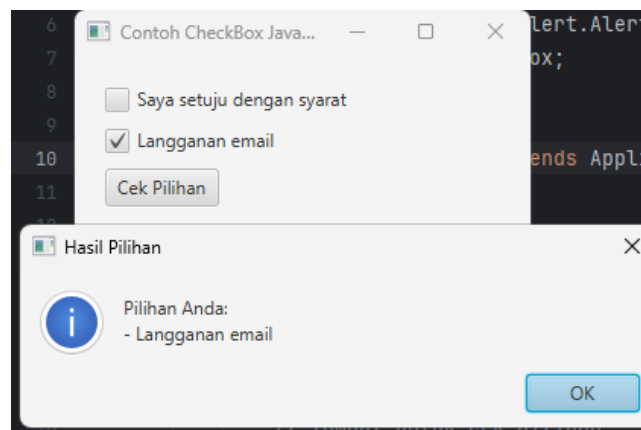
```
// Membuat CheckBox
CheckBox setuju = new CheckBox("Saya setuju dengan syarat");
CheckBox langganan = new CheckBox("Langganan email");

// Tombol untuk cek pilihan
Button cekButton = new Button("Cek Pilihan");

cekButton.setOnAction(e -> {
    String hasil = "Pilihan Anda:\n";
    hasil += setuju.isSelected() ? "- Setuju dengan syarat\n" : "";
    hasil += langganan.isSelected() ? "- Langganan email\n" : "";
    if (!setuju.isSelected() && !langganan.isSelected()) {
        hasil += "Tidak ada pilihan yang dipilih.";
    }

    // Menampilkan hasil di dialog alert
    Alert alert = new Alert(AlertType.INFORMATION);
    alert.setTitle("Hasil Pilihan");
    alert.setHeaderText(null);
    alert.setContentText(hasil);
    alert.showAndWait();
});
```

Pengguna bisa centang satu, dua, atau tidak sama sekali. Cocok untuk pilihan bebas.



<https://jenkov.com/tutorials/javafx/checkbox.html>

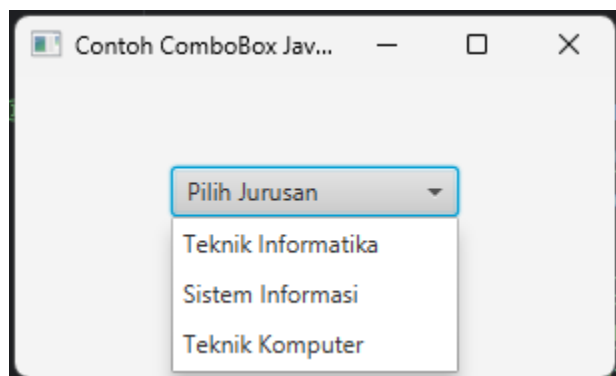
ComboBox (Pilihan dari Daftar Dropdown)

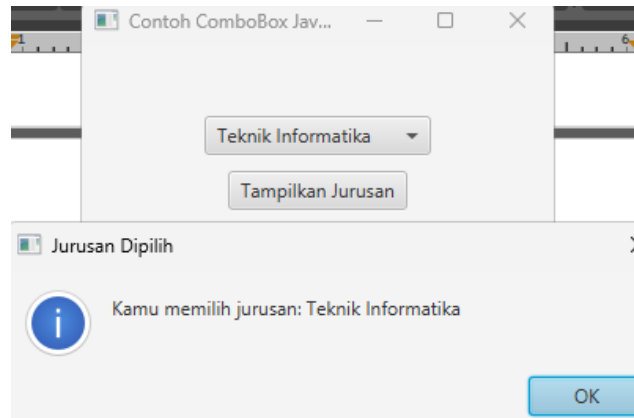
ComboBox adalah menu drop-down yang bisa diklik dan berisi daftar pilihan. Contohnya: memilih jurusan.

```
// Membuat ComboBox dengan daftar jurusan
ComboBox<String> comboJurusan = new ComboBox<>();
comboJurusan.getItems().addAll(
    "Teknik Informatika",
    "Sistem Informasi",
    "Teknik Komputer"
);
comboJurusan.setPromptText("Pilih Jurusan");

// Tombol untuk menampilkan jurusan yang dipilih
Button btnPilih = new Button("Tampilkan Jurusan");
btnPilih.setOnAction(e -> {
    String jurusan = comboJurusan.getValue();
    if (jurusan != null) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Jurusan Dipilih");
        alert.setHeaderText(null);
        alert.setContentText("Kamu memilih jurusan: " + jurusan);
        alert.showAndWait();
    } else {
        Alert alert = new Alert(Alert.AlertType.WARNING);
        alert.setTitle("Peringatan");
        alert.setHeaderText(null);
        alert.setContentText("Silakan pilih jurusan terlebih dahulu!");
        alert.showAndWait();
    }
});
```

Pengguna cukup klik ComboBox dan pilih salah satu jurusan dari daftar.





<https://jenkov.com/tutorials/javafx/combobox.html>

4.4 APLIKASI DATA MAHASISWA (JavaFX)

Kamu akan membuat aplikasi sederhana yang memiliki form input data mahasiswa menggunakan JavaFX. Form ini berisi:

- **TextField** untuk mengisi Nama dan NIM
- **RadioButton** untuk memilih Jenis Kelamin ("Laki-laki" atau "Perempuan")
- **ComboBox** untuk memilih Jurusan (misalnya: Teknik Informatika, Sistem Informasi, dll)
- **Button** "Simpan" untuk menampilkan data yang diinput

Bagaimana cara kerjanya di JavaFX?

1. Buat **Scene** dengan layout (biasanya VBox atau GridPane) sebagai tempat menata komponen.
2. Tambahkan TextField untuk Nama dan NIM.
3. Tambahkan ToggleGroup yang berisi dua RadioButton untuk jenis kelamin. ToggleGroup memastikan hanya satu pilihan yang aktif.
4. Tambahkan ComboBox berisi daftar jurusan.
Tambahkan tombol Button "Simpan".
Buat event handler untuk tombol, yang ketika ditekan:
 - Mengambil data dari TextField, RadioButton, dan ComboBox.
 - Menampilkan data tersebut dalam dialog (pakai Alert).

Contoh Kode Singkat:

```
import javafx.application.Application;
import javafx.scene.Scene;
import javafx.scene.control.*;
import javafx.scene.layout.GridPane;
import javafx.stage.Stage;

public class MyJavaFXApps extends Application {
    @Override
    public void start(Stage stage) {
        // Membuat komponen input
        TextField namaField = new TextField();
        TextField nimField = new TextField();

        RadioButton rbLaki = new RadioButton("Laki-laki");
        RadioButton rbPerempuan = new RadioButton("Perempuan");
        ToggleGroup genderGroup = new ToggleGroup();
        rbLaki.setToggleGroup(genderGroup);
        rbPerempuan.setToggleGroup(genderGroup);

        ComboBox<String> jurusanBox = new ComboBox<>();
        jurusanBox.getItems().addAll("Teknik Informatika", "Sistem Informasi", "Teknik Elektro");

        Button simpanButton = new Button("Simpan");
```

```
// Layout pakai GridPane
GridPane grid = new GridPane();
grid.setVgap(10);
grid.setHgap(10);

grid.add(new Label("Nama:"), 0, 0);
grid.add(namaField, 1, 0);
grid.add(new Label("NIM:"), 0, 1);
grid.add(nimField, 1, 1);
grid.add(new Label("Jenis Kelamin:"), 0, 2);
grid.add(rbLaki, 1, 2);
grid.add(rbPerempuan, 2, 2);
grid.add(new Label("Jurusan:"), 0, 3);
grid.add(jurusanBox, 1, 3);
grid.add(simpanButton, 1, 4);

// Event handler tombol simpan
simpanButton.setOnAction(e -> {
    String nama = namaField.getText();
    String nim = nimField.getText();
    RadioButton selectedGender = (RadioButton) genderGroup.getSelectedToggle();
    String gender = (selectedGender != null) ? selectedGender.getText() : "Belum pilih";
    String jurusan = jurusanBox.getValue();
```



```

        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Data Mahasiswa");
        alert.setHeaderText("Informasi Data");
        alert.setContentText("Nama: " + nama + "\n" +
            "NIM: " + nim + "\n" +
            "Jenis Kelamin: " + gender + "\n" +
            "Jurusan: " + jurusan);
        alert.showAndWait();
    });

    Scene scene = new Scene(grid, 400, 250);
    stage.setTitle("Form Data Mahasiswa");
    stage.setScene(scene);
    stage.show();
}

public static void main(String[] args) {
    launch(args);
}
}

```

Contoh Output:

The screenshot displays a Java Swing application with two windows. The main window, titled "Form Data Mahasiswa", contains a form with the following fields and controls:

- Nama:** A text field containing "Wira Yudha Aji Pratama".
- NIM:** A text field containing "202310370311010".
- Jenis Kelamin:** Two radio buttons, "Laki-laki" (selected) and "Perempuan".
- Jurusan:** A dropdown menu showing "Teknik Informatika".
- Simpan:** A button to save the data.

An information dialog box, titled "Data Mahasiswa", is overlaid on the main window. It has a header "Informasi Data" and a blue information icon. The dialog displays the entered data:

```

Nama: Wira Yudha Aji Pratama
NIM: 202310370311010
Jenis Kelamin: Laki-laki
Jurusan: Teknik Informatika

```

An "OK" button is located at the bottom right of the dialog box.

4.5 TABEL (JavaFX)

Untuk membuat **tabel di JavaFX**, kamu bisa menggunakan **TableView**, yaitu komponen GUI yang digunakan untuk menampilkan data dalam bentuk tabel.

Contoh cara membuat TableView:

Class Mahasiswa:

```
public class Mahasiswa {  
    private String nama;  
    private int umur;  
    private double tinggiBadan;  
  
    public Mahasiswa(String nama, int umur, double tinggiBadan) {  
        this.nama = nama;  
        this.umur = umur;  
        this.tinggiBadan = tinggiBadan;  
    }  
  
    public String getNama() {  
        return nama;  
    }  
  
    public int getUmur() {  
        return umur;  
    }  
  
    public double getTinggiBadan() {  
        return tinggiBadan;  
    }  
}
```

Class Utama:

```

@Override
public void start(Stage primaryStage) {
    TableView<Mahasiswa> table = new TableView<>();

    // Kolom nama
    TableColumn<Mahasiswa, String> namaCol = new TableColumn<>("Nama");
    namaCol.setCellValueFactory(new PropertyValueFactory<>("nama"));

    // Kolom umur
    TableColumn<Mahasiswa, Integer> umurCol = new TableColumn<>("Umur");
    umurCol.setCellValueFactory(new PropertyValueFactory<>("umur"));

    // Kolom tinggi badan
    TableColumn<Mahasiswa, Double> tinggiCol = new TableColumn<>("Tinggi Badan");
    tinggiCol.setCellValueFactory(new PropertyValueFactory<>("tinggiBadan"));

    // Tambahkan kolom ke tabel
    table.getColumns().addAll(namaCol, umurCol, tinggiCol);

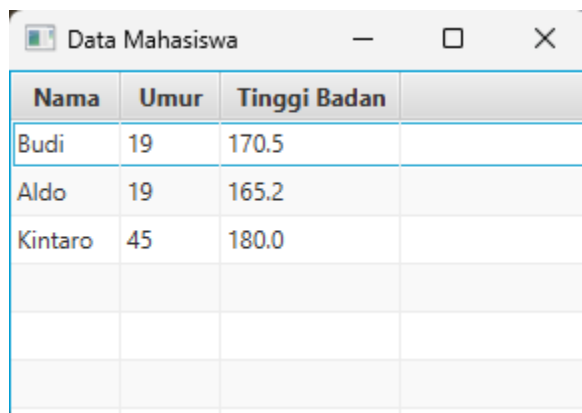
    // Data (ArrayList → ObservableList)
    ObservableList<Mahasiswa> data = FXCollections.observableArrayList(
        new Mahasiswa("Budi", 19, 170.5),
        new Mahasiswa("Aldo", 19, 165.2),
        new Mahasiswa("Kintaro", 45, 180.0)
    );

    // Isi tabel
    table.setItems(data);

    // Tampilkan di layout
    VBox root = new VBox(table);
    Scene scene = new Scene(root, 400, 300);
    primaryStage.setTitle("Data Mahasiswa");
    primaryStage.setScene(scene);
    primaryStage.show();
}

```

Output:



Nama	Umur	Tinggi Badan	
Budi	19	170.5	
Aldo	19	165.2	
Kintaro	45	180.0	

Penjelasan:

```
TableView<Mahasiswa> table = new TableView<>();
```

berarti kamu **membuat sebuah tabel di JavaFX** yang dirancang khusus untuk menampilkan **data bertipe Mahasiswa**.

```
TableColumn<Mahasiswa, String> namaCol = new TableColumn<>("Nama");
namaCol.setCellValueFactory(new PropertyValueFactory<>("nama"));
```

TableColumn<Mahasiswa, String> namaCol:

- Berarti, Kolom ini nanti menampilkan data dari objek Mahasiswa. Kolom ini bertipe String, karena nama bertipe String.

new TableColumn<>("Nama"):

- Membuat kolom baru dengan **judul** di atas kolom yaitu "Nama".

setCellValueFactory(new PropertyValueFactory<>("nama")):

- Memberitahu JavaFX bahwa isi dari kolom ini diambil dari **getNama()** di dalam objek Mahasiswa.
- PropertyValueFactory akan otomatis mencari **getter** bernama getNama() sesuai string "nama".

```
table.getColumns().addAll(namaCol, umurCol, tinggiCol);
```

berfungsi untuk **menambahkan semua kolom ke dalam TableView**, agar nanti bisa ditampilkan ke layar.

```
ObservableList<Mahasiswa> data = FXCollections.observableArrayList(
    new Mahasiswa("Budi", 19, 170.5),
    new Mahasiswa("Aldo", 19, 165.2),
    new Mahasiswa("Kintaro", 45, 180.0)
);
```

ObservableList adalah **versi spesial dari List di JavaFX** yang bisa **memberi tahu (notify)** GUI secara otomatis jika isinya berubah. Ini sangat penting saat kamu bekerja dengan komponen seperti TableView, ListView, atau ComboBox, karena mereka **butuh tahu kapan data berubah** supaya bisa langsung memperbarui tampilannya. Mirip seperti ArrayList. Bedanya, ArrayList tidak bisa memberi tahu GUI secara otomatis jika isinya berubah.

Baris ini membuat 3 objek mahasiswa beserta attribute nya, menaruhnya ke dalam ObservableList, dan menyimpannya ke variabel "data". Nantinya data ini dipasang ke tabel seperti ini:

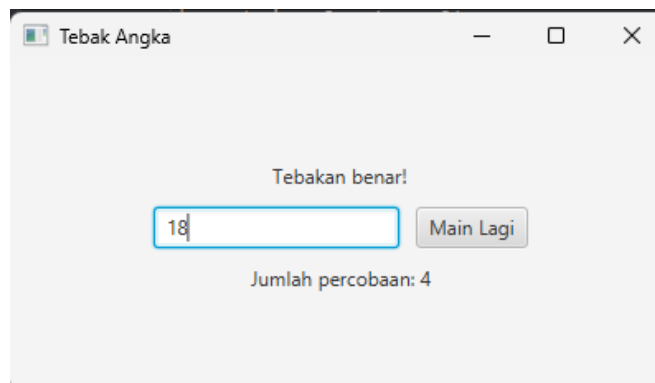
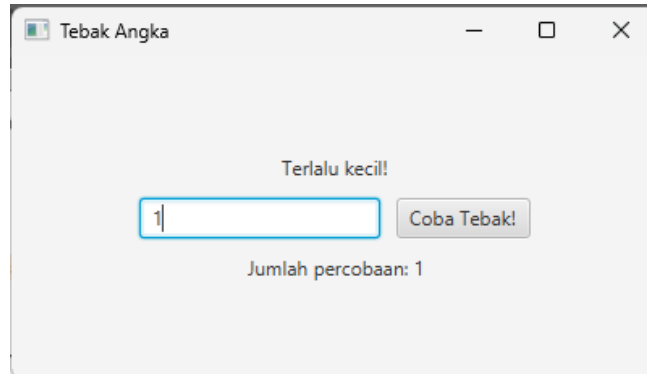
```
table.setItems(data);
```

BAB V: CODELAB & TUGAS

5.1 CODELAB

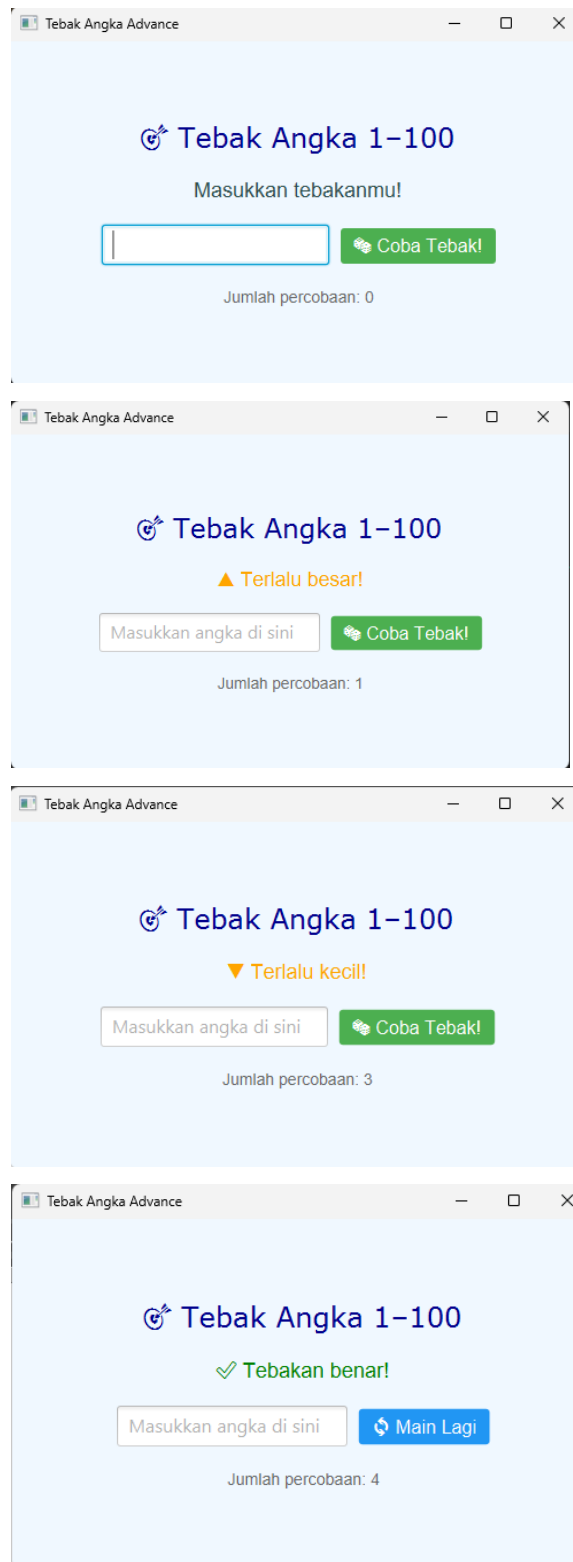
Kriteria Umum

1. Komputer memilih angka secara acak dari 1–100 saat aplikasi dibuka.
2. User diberi:
 - Satu TextField untuk mengetik tebakan.
 - Satu Button bertuliskan **"Coba Tebak!"**.
 - Satu Label untuk memberi feedback: "Terlalu besar!", "Terlalu kecil!", atau "Tebakan benar!"
 - Satu Label tambahan menunjukkan **jumlah percobaan**.
3. Jika jawaban benar:
 - Tombol berubah menjadi **"Main Lagi"** dan angka baru di-generate otomatis.
4. Gunakan layout campuran seperti VBox dan HBox.
5. Contoh output:



Kriteria Tambahan

Buat tampilan UI nya sama seperti gambar dibawah ini:



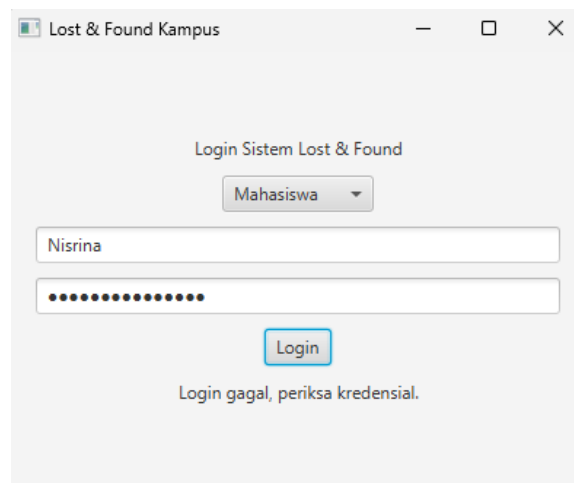
Notes:

Gunakan **JavaFX** untuk membuat tampilan GUI di **IntelliJ**.

Bagi yang codelabnya tidak sempat dinilai oleh asisten di lab pada saat waktu praktikum, silahkan buat laporan yang menjelaskan alur” pembuatan codelab diatas. Di zip kemudian kumpulkan di llab sebelum deadline.

5.2 TUGAS

Dengan logic yang sama, kembangkanlah program dari tugas di modul 5 sebelumnya yang awalnya CLI menjadi GUI menggunakan JavaFX seperti pada contoh output di bawah ini:

Tampilan Login:**Tampilan Dashboard Mahasiswa:**

Lost & Found Kampus

Selamat datang, Herdiana
Laporkan Barang Hilang/Temuan

Daftar Laporan Anda

Nama	Lokasi
HP	Meja A-01

Tampilan Dashboard Admin:

Lost & Found Kampus

Halo, Administrator admin

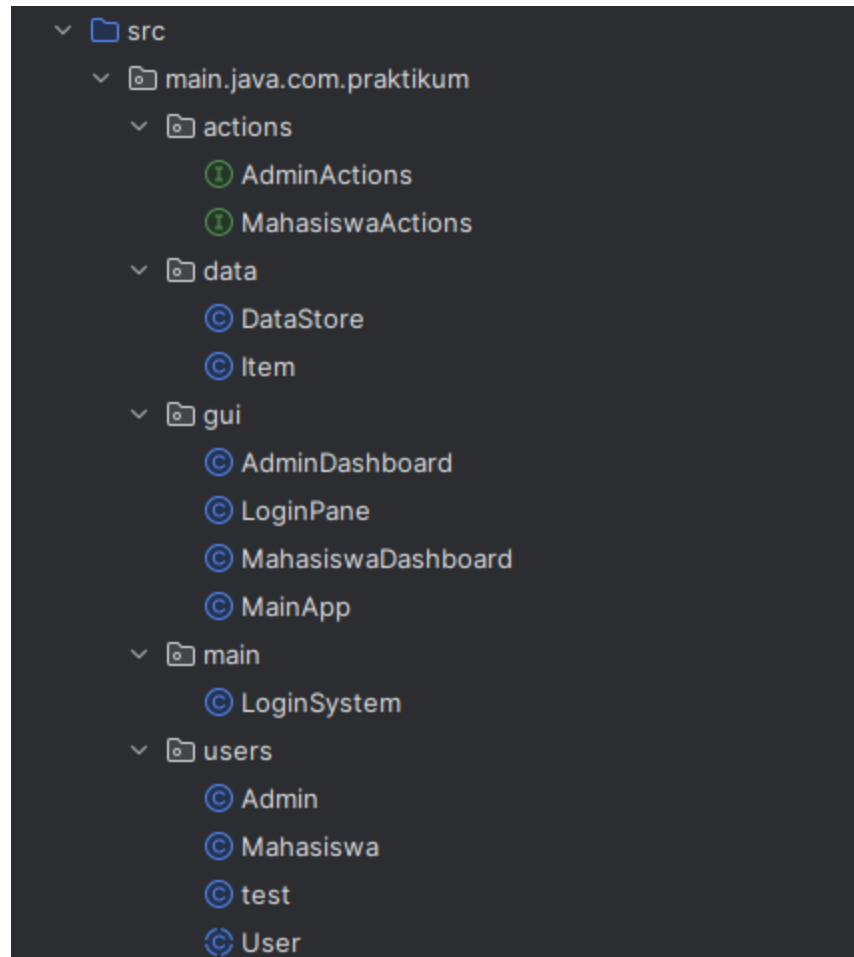
Laporan Barang

Nama	Lokasi	Status
HP	Meja A15	Claimed
Dompel	Meja B-19	Claimed
Wdaef	ev	Claimed
HP	Meja A-01	Reported

Data Mahasiswa

Nama	NIM
Ken Aryo	202310370311006
Wira Yudha	202310370311010
Nisrrina	202310370311321
Herdiana	202310370311129

Struktur Package:

**Tips:**

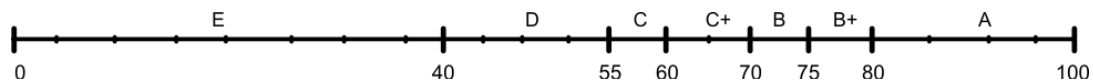
Dokumentasi untuk membuat tabel dapat dilihat [disini](#). Silahkan kembangkan kemampuan untuk membaca dokumentasi. Skill membaca dokumentasi merupakan kemampuan dasar yang harus dimiliki oleh developer!

5.3 KRITERIA PENILAIAN

Aspek Penilaian	Poin
CODELAB	Total 20%
Sesuai Kriteria Umum	10%
Membuat Kriteria Tambahan	10%
TUGAS	Total 80%

Ketepatan kode & output. Contoh: Untuk logic, sama seperti tugas modul 5. Untuk tampilan sesuaikan dengan GUI pada contoh tugas modul 6 ini.	20%
Kemampuan menjawab pertanyaan. Contoh pertanyaan: 1. Apa itu label? 2. Apa itu textfield? 3. Apa itu radiobox?	20%
Kemampuan menyelesaikan challenge. Contoh challenge: 1. Buat label yang bertuliskan 'Nama'. 2. Buat textfield baru. 3. Buat radiobox dengan opsi A,B,C.	20%
Kemampuan menjelaskan alur kode. Contoh: Pertama, kode menjalankan ini.. Kemudian ini... dan seterusnya.	20%

5.4 SKALA PENILAIAN



A = (81 - 100) → **Sepuh**

B+ = (75 - 80) → **Sangat baik**

B = (70 - 74) → **Baik**

C+ = (60 - 69) → **Cukup baik**

C = (55 - 59) → **Cukup**

D = (41 - 54) → **Kurang**

E = (0 - 40) → **Bro really...**

5.5 SUMMARY AKHIR MODUL

Akhirnya, kita berhasil menamatkan... (kan)... (kan)... Praktikum PBO di Semester 2 🎉. Silahkan foto tulisan ini, up di SG kalian tag akun @LabitUMM pakai lagu ini: <https://www.youtube.com/watch?v=13ARO0HDZsQ>. Silahkan pamer nilai kalian juga (kalau A semua wkwk).

Terima kasih buat kalian yang sudah serius, sabar, dan tetap semangat walau harus debug berjam-jam, ngoding sambil ngerjain tugas lain, dan kadang chat asisten jam 12 malam 😊.

Semoga semua ilmu dan pengalaman selama praktikum ini bisa jadi bekal buat semester depan dan masa depan. Yang penting bukan cuma bisa ngoding, tapi bisa *bertahan* saat error, bisa *berpikir* saat stuck, dan bisa *belajar terus* meski capek. Karena jadi programmer (atau apapun nanti kalian pilih) itu bukan soal hasilnya, tapi prosesnya.

See you di praktikum selanjutnya. Jangan lupa jaga kesehatan, jaga solidaritas, dan tetap jadi versi terbaik dari diri kalian sendiri 💪🔥

[#PBO2025](#)

[#LabiTUMM](#)

[#NgodingSantuy](#)

[#KeluarDariWhileTrue](#)

Oh iya, jangan lupa belajar untuk UAP.