Dasar-Dasar Pemrograman 2

Lab 09

GUI

I. Sejarah

Java mempunyai tiga macam library *Graphical User Interface* (GUI), yang sebenarnya hanyalah kumpulan *class* dalam JAVA API yang dapat kita gunakan.

a. *Abstract Window Toolkit* (AWT) sudah menjadi bagian dari Java sejak awal, tahun 1996. AWT bekerja dengan cara mem-*passing* panggilan-panggilan *"native"* pada sistem operasi kita. Jadi, kalau kita membuat sebuah *checkbox* pada AWT, kita sebenarnya memberi tahu sistem operasi kita untuk membuat sebuah *checkbox*. Semua *class-class* AWT berada pada *package* `java.awt`.

b. Kelemahan dari AWT adalah program kita akan terlihat berbeda pada sistem operasi yang berbeda, sebuah *checkbox* pada Linux akan berbeda dengan sebuah *checkbox* pada Windows. Hal ini membuat adanya kesulitan dalam membuat *layout* program yang konsisten. Berangkat dari permasalahan tersebut, Swing ditambahkan pada Java pada tahun 1998. Swing tidak memerintahkan sistem operasi untuk membuat sebuah *checkbox*, melainkan membuat *checkbox* itu sendiri. Dengan itu, *checkbox* tersebut akan konsisten pada sistem operasi apapun. *Class-class* Swing terdapat pada package `javax.swing`. Karena Swing dikembangkan di atas *class* AWT, kita tetap dapat melihat Swing menggunakan *class-class* dari *package* `java.awt` juga.

c. JavaFX diperkenalkan pertama kali sebagai sebuah *library* eksternal pada tahun 2008, dan ditambahkan ke Java pada 2014. JavaFX lebih berfokus pada fitur-fitur GUI modern seperti animasi, CSS *styling,* serta mengandalkan *graphic card* dari komputer untuk menangani *rendering*. *Class-class* JavaFX berada pada package `javafx`, dan dalam file `.jar` java.

Walaupun JavaFX merupakan *library* GUI yang terbaru, kita akan lebih berfokus pada Swing karena:

- Swing adalah cara yang baik untuk lebih membiasakan diri dengan konsep OOP, *inheritance,* dan alur umum program. Jadi walaupun tujuan akhir kita

bukanlah pemrograman dengan Swing, kita dapat mempelajari banyak hal dengan membiasakan diri dengan Swing!

- Salah satu keuntungan lain dari Swing adalah Swing sudah ada untuk kurun waktu yang lama. Sehingga, jika kita bertanya-tanya tentang sesuatu, kemungkinan besar pertanyaan serupa sudah ditanyakan pada Stack Overflow! 😁

II. JFrame

Dalam membuat GUI menggunakan Java Swing, hal pertama yang harus dilakukan adalah menampilkan sebuah *window* yang menjadi *interface* dimana *user* dapat berinteraksi. Hal ini menjadi perbedaan utama dibanding saat kita menggunakan *Command Line Interface* (CLI). Di Java Swing, kita menggunakan JFrame untuk melakukan hal tersebut. Perhatikan potongan kode berikut!

```
import javax.swing.JFrame;

public class MyGui{

      public static void main(String[] args){
            JFrame frame = new JFrame("Coba-Coba Swing");
            frame.setLocationRelativeTo(null);
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            frame.setSize(300, 300);
            frame.setVisible(true);
      }
}
```

Kode `JFrame frame = new JFrame("Happy Coding");` adalah *constructor* JFrame. Di dalamnya, kita dapat mem-*pass* sebuah *string* yang akan menjadi judul dari *window* yang akan kita buat. Selain itu, kita juga dapat memberi judul *window* dengan menggunakan *setter* seperti `frame.setTitle("Swing Coba-Coba");`

Kode `frame.setLocationRelativeTo(null);` akan membuat frame muncul di tengah-tengah layar kita.

Kode `frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);` akan membuat program berhenti saat frame tersebut kita tutup. Selain `EXIT_ON_CLOSE`, terdapat beberapa pilihan lain seperti `DO_NOTHING_ON_CLOSE`, `HIDE_ON_CLOSE`, `DISPOSE_ON_CLOSE` yang perbedaan lebih detailnya dapat kamu cari di sumber lain maupun IntelliJ.

Kode `frame.setSize(300, 300);` digunakan untuk mengatur ukuran dari frame. Urutan parameternya adalah *width* lalu *height*.

Dan terakhir, kode `frame.setVisible(true);` berguna untuk menampilkan frame yang telah kita buat ke layar.

III. JButton

Dalam membuat GUI, sering kali kita membutuhkan sebuah tombol yang dapat diklik dan tentunya melakukan sesuatu saat diklik. Untuk membuatnya, kita dapat menggunakan JButton di Java Swing. Perhatikan potongan kode berikut untuk membuat dan meletakkannya dalam frame yang tadi telah kita buat!

```
JButton button = new JButton("Click me!");
frame.add(button);
```

Kode di atas akan membuat sebuah tombol bertuliskan “Click me!”, lalu meletakkannya ke frame yang sebelumnya telah kita buat. Selain dapat mengatur tulisan pada tombol, JButton juga memiliki *constructor* lain yang memungkinkan kita untuk meletakkan *icon*, atau *icon* dan tulisan secara bersamaan pada tombol tersebut.

Untuk melakukan sesuatu saat tombol diklik, kita dapat menambahkan *event listener* yang lebih lengkapnya akan dibahas di bagian VI dokumen ini.

IV. JLabel

JLabel akan kita gunakan untuk membuat tulisan yang tidak dapat diubah oleh user. Selain mengatur teks yang akan ditampilkan, kita juga dapat mengatur alignment teks tersebut dengan beberapa pilihan yang disediakan Java Swing seperti LEFT, CENTER, dan RIGHT. Berikut adalah contoh penggunaan JLabel.

```
JLabel label = new JLabel("Hello world!", SwingConstants.CENTER);
frame.add(label);
```

Alignment juga dapat diubah dengan menggunakan *method* `setHorizontalAlignment` yang dimiliki oleh JLabel.

V. Layout Manager

Layout Managers pada Java menyediakan tingkatan *abstraction* yang secara otomatis memetakan *user interface* (UI) pada semua *window system.* Komponen-komponen UI dimuat pada *container-container*, dan setiap *container* tersebut akan menganut sebuah Layout Manager yang berperan untuk mengatur setiap komponen UI.

Layout Manager ditambahkan pada *container* menggunakan *method* `setLayout(LayoutManager);`

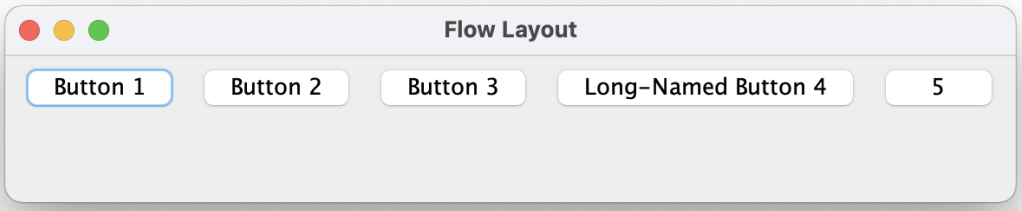
A. FlowLayout

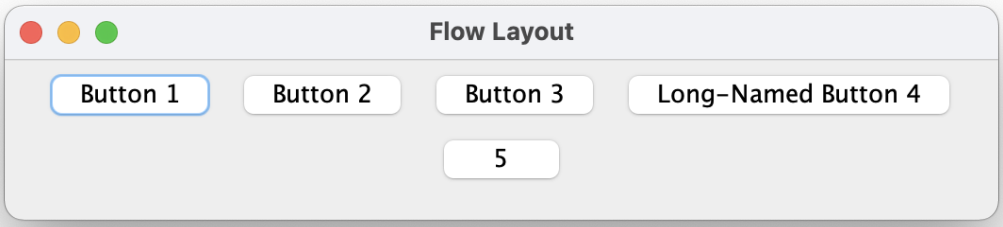
FlowLayout adalah *layout manager* yang *default* bagi semua JPanel. FlowLayout menaruh semua komponen pada satu baris dan memulai baris baru apabila baris yang sebelumnya sudah tidak dapat memuat komponen baru.

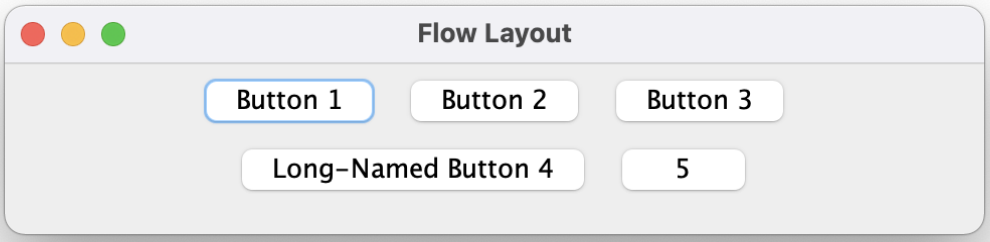
```
JPanel flowLayoutExperiment = new JPanel();
flowLayoutExperiment.setLayout(new FlowLayout());

flowLayoutExperiment.add(new JButton("Button 1"));
flowLayoutExperiment.add(new JButton("Button 2"));
flowLayoutExperiment.add(new JButton("Button 3"));
flowLayoutExperiment.add(new JButton("Long-Named Button 4"));
flowLayoutExperiment.add(new JButton("5"));

frame.add(flowLayoutExperiment);
```







B. GridLayout

GridLayout akan membuat semua komponen menjadi seukuran dan menampilkannya dalam baris dan kolom sesuai permintaan.

```
JPanel gridLayoutExperiment = new JPanel();
gridLayoutExperiment.setLayout(new GridLayout(2,3));

gridLayoutExperiment.add(new JButton("Button 1"));
gridLayoutExperiment.add(new JButton("Button 2"));
gridLayoutExperiment.add(new JButton("Button 3"));
gridLayoutExperiment.add(new JButton("Long-Named Button 4"));
gridLayoutExperiment.add(new JButton("5"));

frame.add(gridLayoutExperiment);
```

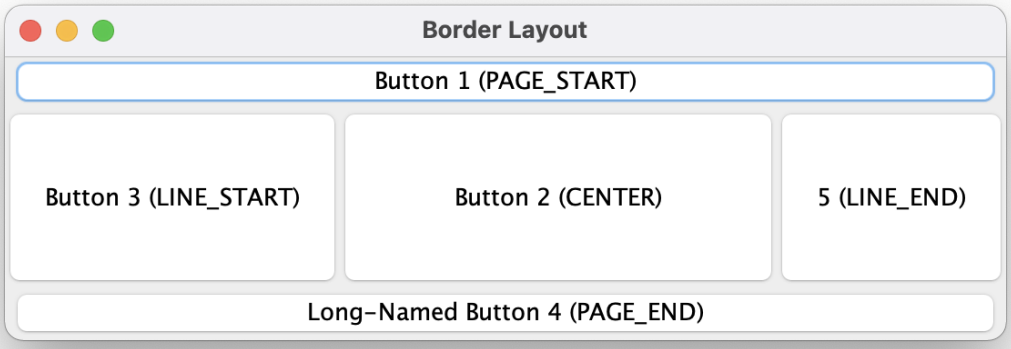
C. BorderLayout

Setiap *content pane* diinisialisasi untuk menggunakan BorderLayout. BorderLayout menaruh komponen pada 5 area utama: *top, bottom, left, right,* dan *centre*. Semua sisa ruangan akan dimuat di area *centre.*

```
JPanel borderLayoutExperiment = new JPanel();
borderLayoutExperiment.setLayout(new BorderLayout());

borderLayoutExperiment.add(new JButton("Button 1 (PAGE_START)"),
BorderLayout.PAGE_START);
borderLayoutExperiment.add(new JButton("Button 2 (CENTER)"),
BorderLayout.CENTER);
borderLayoutExperiment.add(new JButton("Button 3 (LINE_START)"),
BorderLayout.LINE_START);
borderLayoutExperiment.add(new JButton("Long-Named Button 4 (PAGE_END)"),
BorderLayout.PAGE_END);
borderLayoutExperiment.add(new JButton("5 (LINE_END)"),
BorderLayout.LINE_END);

frame.add(borderLayoutExperiment);
```



D. BoxLayout

BoxLayout menaruh semua komponen di satu baris atau satu kolom yang sama, mengikuti ukuran maksimum masing-masing komponen.

```
JPanel boxLayoutExperiment = new JPanel();
boxLayoutExperiment.setLayout(new
BoxLayout(boxLayoutExperiment, BoxLayout.Y_AXIS));

JButton button;

button = new JButton(("Button 1"));
button.setAlignmentX(Component.CENTER_ALIGNMENT);
boxLayoutExperiment.add(button, boxLayoutExperiment);
```

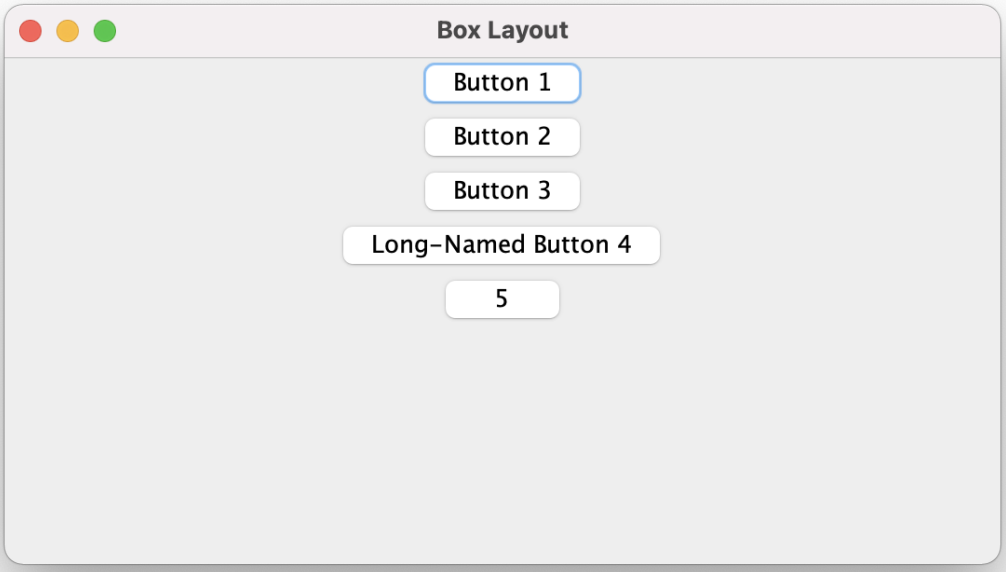
```
button = new JButton(("Button 2"));
button.setAlignmentX(Component.CENTER_ALIGNMENT);
boxLayoutExperiment.add(button, boxLayoutExperiment);

button = new JButton(("Button 3"));
button.setAlignmentX(Component.CENTER_ALIGNMENT);
boxLayoutExperiment.add(button, boxLayoutExperiment);

button = new JButton(("Long-Named Button 4"));
button.setAlignmentX(Component.CENTER_ALIGNMENT);
boxLayoutExperiment.add(button, boxLayoutExperiment);

button = new JButton(("5"));
button.setAlignmentX(Component.CENTER_ALIGNMENT);
boxLayoutExperiment.add(button, boxLayoutExperiment);

frame.add(boxLayoutExperiment);
```



VI. Event Listeners

Sejauh ini, hal-hal yang telah kita lakukan hanyalah menampilkan komponen-komponen Java Swing dan tidak ada interaksi apapun. Saat menekan tombol yang telah kita buat dengan JButton misalnya, tak ada perubahan atau hal lain yang terjadi di GUI yang kita buat. Nah, untuk melakukan sesuatu saat JButton tadi diklik, kita dapat menambahkan hal yang dinamakan ***event listener***.

Event listener merupakan sebuah *object* yang di dalamnya terdapat *method* yang akan dipanggil saat sebuah *event* tertentu muncul. *Event* ini dapat berupa banyak hal seperti saat JButton diklik oleh *user*, *user* menekan tombol tertentu di *keyboard*, atau saat *user* mengklik *mouse*-nya.

Contoh pembuatan *event listener* adalah sebagai berikut:

```
import java.awt.event.*;

public class SimpleActionListener implements ActionListener{

      public void actionPerformed(ActionEvent event){
            System.out.println("Clicked!");
      }
}
```

Pada potongan kode di atas, kita membuat sebuah *concrete class* SimpleActionListener yang mengimplementasi *interface* ActionListener. Dengan menggunakan *interface* ini, saat sebuah JButton diklik oleh *user*, maka *method* actionPerformed secara otomatis akan dijalankan. *Method* ini tentu dapat kalian buat sesuka hati kalian. Hal yang perlu kita lakukan selanjutnya adalah memasang SimpleActionListener ini ke JButton yang kita mau. Lengkapnya dapat dilihat sebagai berikut:

```
import java.awt.event.*;
import javax.swing.*;

public class MyGui{

      public static void main(String[] args){
            JFrame frame = new JFrame("Happy Coding");

frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            JButton button = new JButton("Click me!");
            frame.add(button);

            SimpleActionListener listener = new
SimpleActionListener();
            button.addActionListener(listener);

            frame.setSize(300, 300);
            frame.setVisible(true);
      }
}
```

```
class SimpleActionListener implements ActionListener{

      public void actionPerformed(ActionEvent event){
            System.out.println("Clicked!");
      }
}
```

Dengan kode di atas, saat kita menekan tombol yang ada di GUI, maka program akan mencetak tulisan “Clicked!” di CLI kita. Kalian juga tentunya dapat melibatkan komponen GUI lain untuk berubah saat tombol tersebut ditekan. Ingat bahwa tiap komponen memiliki *setter* yang dapat dipanggil, jadi manfaatkanlah *setter* tersebut dan gunakan di dalam *method* actionPerformed.

Contoh sebelumnya mungkin terlihat agak panjang karena untuk tiap *event listener* kita diharuskan membuat sebuah *class* baru kemudian memasangnya ke sebuah *object* tertentu. Yah begitulah Java, kalau singkat Python namanya. Namun tentu ada cara untuk membuatnya sedikit lebih singkat, yaitu dengan menggunakan *anonymous class* seperti contoh berikut:

```
ActionListener listener = new ActionListener(){

      public void actionPerformed(ActionEvent event){
            System.out.println("Clicked!");
      }
};
button.addActionListener(listener);
```

Atau kita juga dapat memasangnya langsung tanpa harus menyimpan ActionListener ke sebuah variabel. Hasilnya akan menjadi seperti berikut:

```
button.addActionListener(new ActionListener(){

      public void actionPerformed(ActionEvent event){
            System.out.println("Clicked!");
      }
});
```

Beberapa interaksi lain yang dilakukan *user* dan menghasilkan *event* tertentu dapat dilihat di tabel di bawah ini:

User Action	**Source Object**	**Generated Event Type**
Click a button	JButton	ActionEvent

Click a check box	JCheckBox	ActionEvent
		ItemEvent
Click a radio button	JRadioButton	ActionEvent
		ItemEvent
Press return on a text field	JTextField	ActionEvent
Select a new item	JComboBox	ActionEvent
		ItemEvent
Window opened, closed, etc.	Window	WindowEvent
Mouse pressed, released, etc.	Component	MouseEvent
Key released, pressed, etc.	Component	KeyEvent

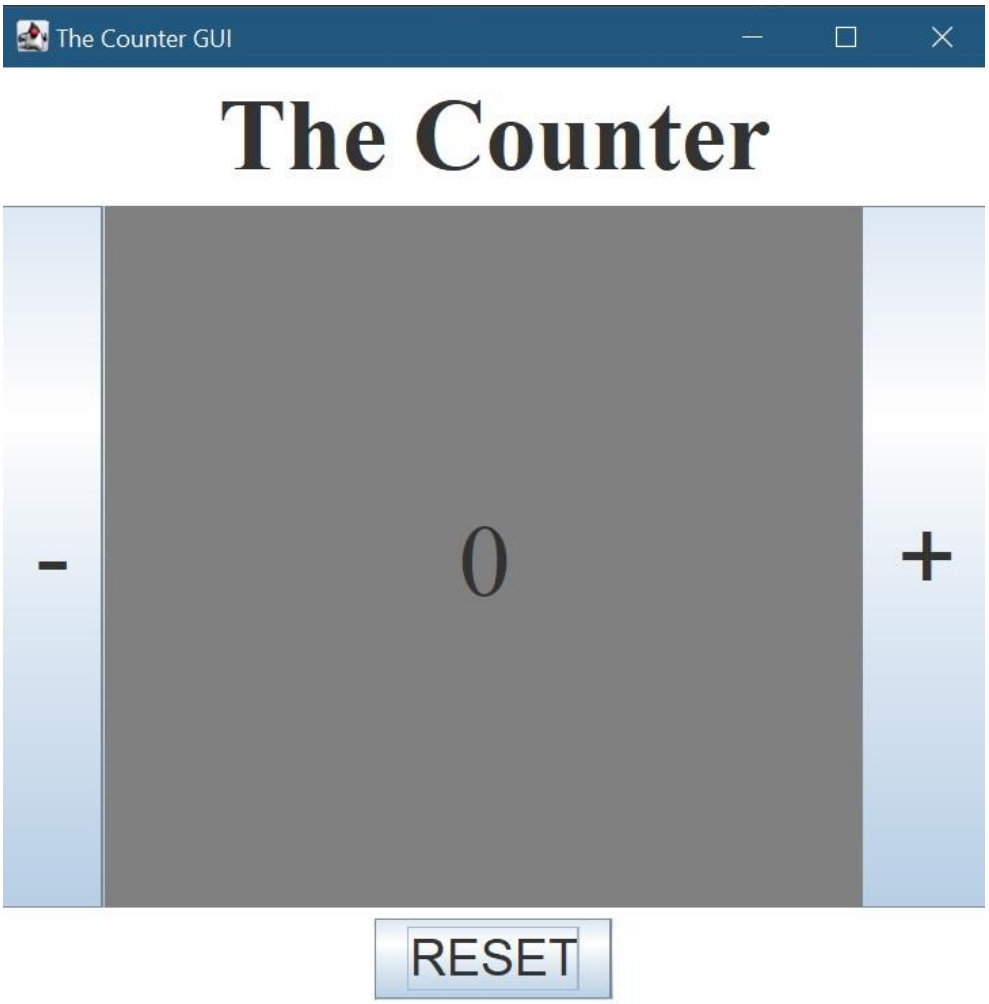
Beberapa *Event* dan *Listener Interface* lain beserta *handler method*-nya dapat juga dilihat di tabel di bawah ini:

Event Class	**Listener Interface**	**Handler Method & Parameter**
ActionEvent	ActionListener	actionPerformed(ActionEvent)
ItemEvent	ItemListener	itemStateChanged(ItemEvent)
WindowEvent	WindowListener	windowClosing(WindowEvent)
		windowOpened(WindowEvent)
		windowIconified(WindowEvent)
		windowDeiconified(WindowEvent)
		windowClosed(WindowEvent)
		windowActivated(WindowEvent)
		windowDeactivated(WindowEvent)
ContainerEvent	ContainerListener	componentAdded(ContainerEvent)
		componentRemoved(ContainerEvent)
MouseEvent	MouseListener	mousePressed(MouseEvent)
		mouseReleased(MouseEvent)
		mouseClicked(MouseEvent)
		mouseExited(MouseEvent)

		mouseEntered(MouseEvent)
KeyEvent	KeyListener	keyPressed(KeyEvent)
		keyReleased(KeyEvent)
		keyTyped(KeyEvent)

Soal Lab 09

The Counter GUI



Kamu merasa senang setelah selesai membuat salinan Dogegochi. Dengan hiburan yang diberikan Dogegochi, kamu bisa rileks dengan mudah dan menikmati liburan. Namun, semua yang baik pasti akan berakhir. Liburan kunjung selesai dan kamu harus balik pada rutinitas kamu sebagai mahasiswa Fasilkom.

Karena liburan yang cukup panjang ini, kamu merasa sedikit sulit untuk kembali terbiasa berkuliah di Fasilkom. Terlalu banyak tugas dan manajemen waktu yang kurang baik menghalangi kamu untuk bisa belajar dengan baik. Sebagai mahasiswa yang teladan, kamu mencari solusi dari masalah ini. Kamu memutuskan untuk membuat program serbaguna **The Counter** yang digunakan sebagai alat pembantu manajemen waktu. Program ini bisa digunakan untuk menghitung dan menyimpan

kesibukan kamu di hari tertentu. Tentu, kamu bisa menambah, mengurangi dan mereset jumlah ini.

Kamu juga terpesona dengan materi GUI sehingga kamu ingin mencoba mengimplementasikan **The Counter** sebagai program berbasis GUI.

Spesifikasi Program

Silahkan buat tampilan GUI sekreatif mungkin. Program tersebut **WAJIB** memiliki spesifikasi sebagai berikut :

- Menggunakan setidaknya salah satu dari FlowLayout, GridLayout, dan/atau BorderLayout
- Memiliki judul frame ‘The Counter GUI’
- Memiliki title ‘The Counter’
- Counter dapat di-increment dengan tombol bertuliskan ‘+’
- Counter dapat di-decrement dengan tombol bertuliskan ‘-’
    - Counter **tidak boleh** bernilai negatif
- Counter dapat di-reset kembali menjadi 0 dengan tombol bertuliskan ‘RESET’
- Desain dari GUI dibebaskan

Selain itu, kamu juga dapat menggunakan bantuan **template** yang telah disediakan di sini.

Komponen Penilaian

- 20% Pembuatan Frame, Title Label, Counter Label, dan Buttons
- 20% Penggunaan setidaknya salah satu dari FlowLayout, GridLayout, dan/atau BorderLayout
- 20% Implementasi increment counter dengan Button ‘+’
- 20% Implementasi decrement counter dengan Button ‘-’
- 20% Implementasi reset counter dengan Button ‘RESET’

Revisi

Belum Ada Revisi

Kumpulkan berkas .java yang telah di-zip dengan format penamaan seperti berikut.
`Lab09_[Kelas]_[KodeAsdos]_[NPM]_[NamaLengkap].zip`
Contoh:
`Lab09_A_LN_1234567890_DekDepe.zip`