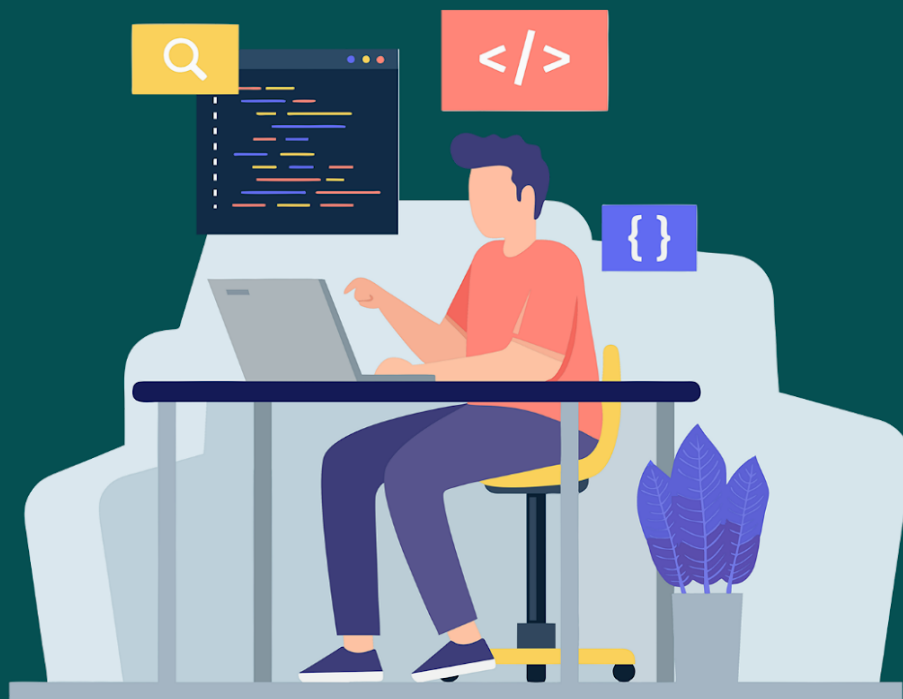


Modul Praktikum  
Desain dan Pemrograman Berorientasi Objek

# INTRODUCTION TO JAVA GUI: APACHE NETBEANS



TIM ASISTEN PEMROGRAMAN  
ANGKATAN 12

Departemen Pendidikan Ilmu Komputer  
Fakultas Pendidikan Matematika dan Ilmu Pengetahuan Alam  
Universitas Pendidikan Indonesia  
2023



## GUI Introduction

GUI stands for Graphical User Interface. It is a type of interface that allows users to interact with a software application or operating system using graphical elements such as icons, buttons, menus, and windows, rather than relying solely on text-based commands. A GUI typically presents the user with a visual representation of the data and controls in a program, allowing them to manipulate and modify it through direct interaction with the interface.

## Java GUI

Java GUI is a type of GUI that allows developers to create interactive user interfaces for Java applications using various libraries and frameworks. Java GUIs can be developed using Swing, JavaFX, AWT (Abstract Window Toolkit), and SWT (Standard Widget Toolkit). Java GUIs provide a powerful and flexible way to create interactive user interfaces for Java applications, and they offer a wide range of libraries and frameworks to choose from.

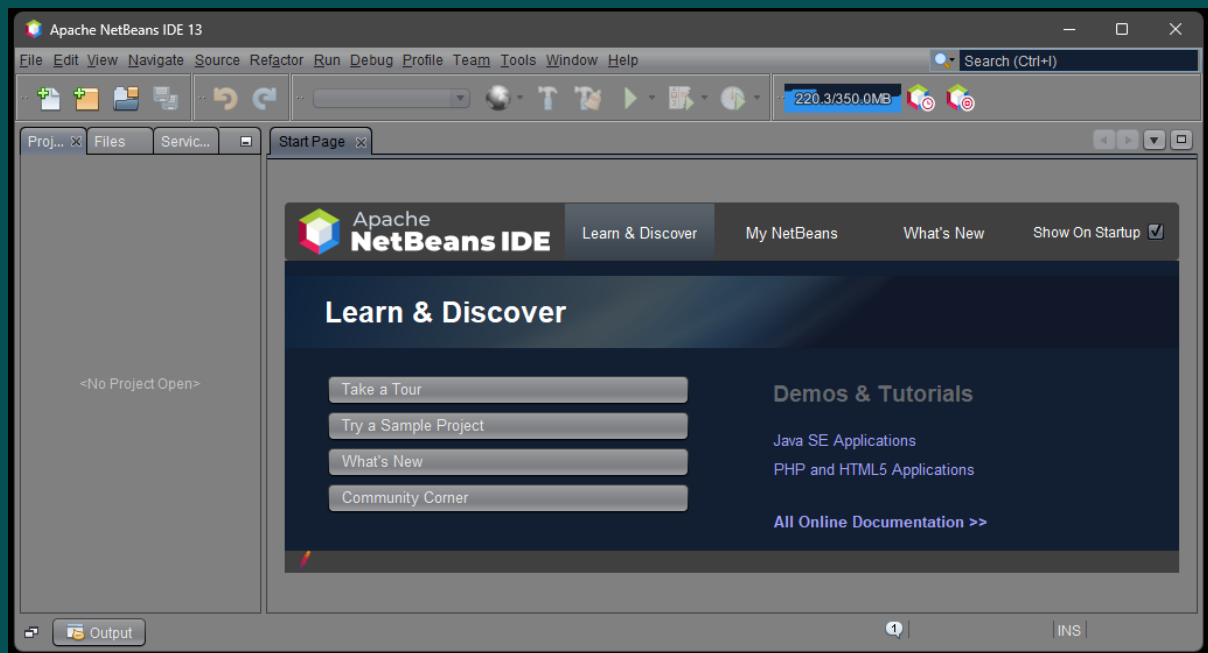
1. **Swing** : Swing is the most commonly used Java GUI library, and it provides a rich set of components such as buttons, labels, menus, and text fields, as well as more complex components such as tables and tree views. Swing also provides support for look and feel customization, which allows developers to customize the appearance of their Java GUI applications.
2. **JavaFX** : JavaFX is a more modern Java GUI framework that was introduced in Java SE 8. It provides a rich set of components, including charts, media players, and 3D shapes, and it is designed to work seamlessly with other Java APIs, such as the Java 2D and Java 3D APIs.
3. **AWT** : AWT is the original Java GUI toolkit, and it provides a low-level API for creating GUI components. AWT is still widely used, particularly for simple GUI applications, but it has been largely superseded by Swing and JavaFX.
4. **SWT** : SWT is a Java GUI toolkit that is designed to provide native look and feel for different platforms. SWT is often used for developing Eclipse plugins and other desktop applications that require a native appearance

Java GUIs can be created using different development tools, such as NetBeans, Eclipse, or IntelliJ IDEA, and they can run on different platforms, including Windows, macOS, and Linux.

## Apache Netbeans

We will learn about **Graphical User Interface (GUI)** in programming with Java using Apache Netbeans. Make sure you already install Apache Netbeans IDE which you can get from their website at <https://netbeans.apache.org/>. I recommend installing Apache Netbeans 12th version or above because that version is more stable. To install the previous version you can get it from this [link](#).

### Welcome to Apache Netbeans IDE!

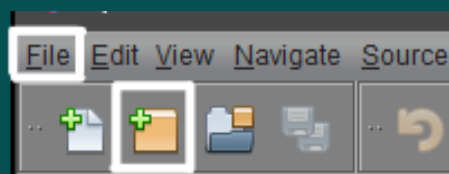


When you first open Apache Netbeans, you will go to Apache Netbeans' start page as you can see in the picture above. After successfully installing Apache Netbeans IDE, we will learn how to create your first project.

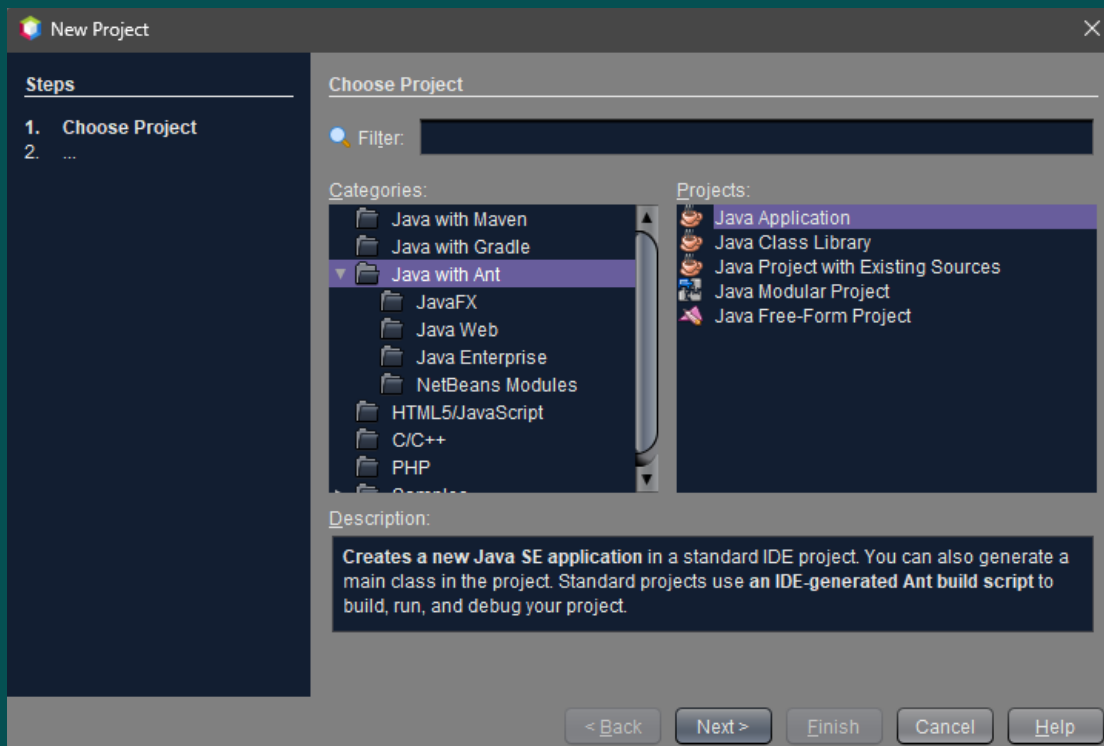
### Setting Up Your Project

Take the steps below to set up a new Java project.

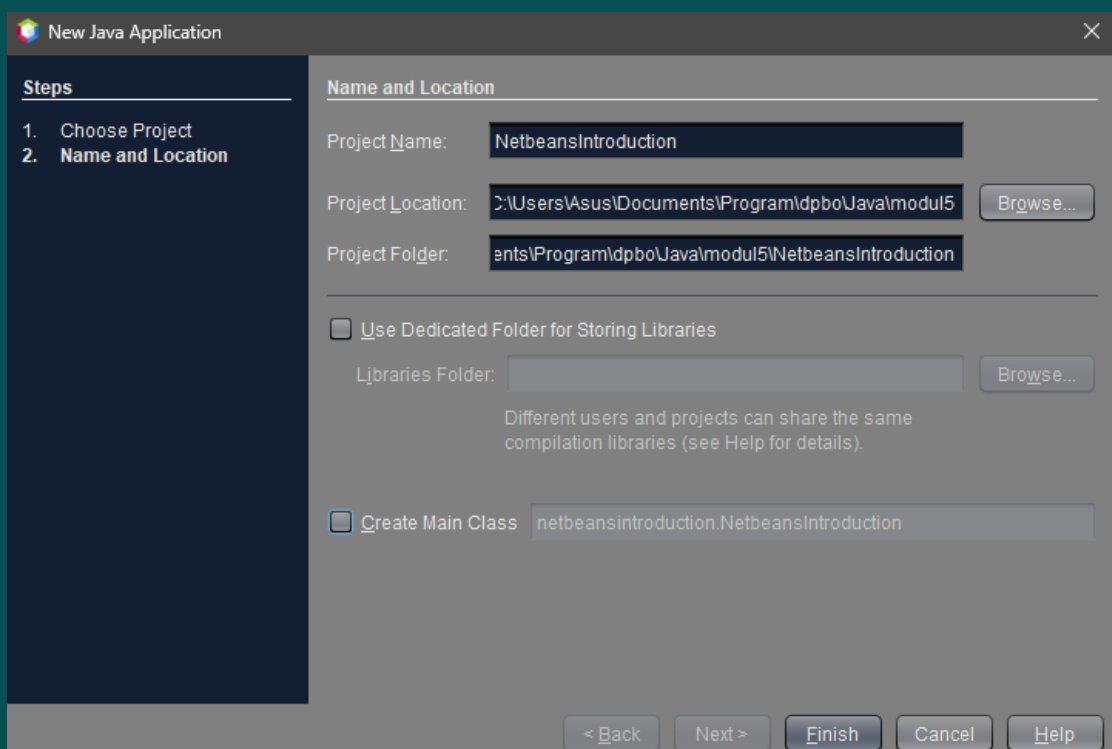
1. Go to **File > New Project** or click the **"New Project"** button in the toolbar or just use **Ctrl + Shift + N**.



2. In the New Project wizard, you can choose **Java With Ant (in Categories Section) > Java Application (in Projects Section)**, and then click **Next**.



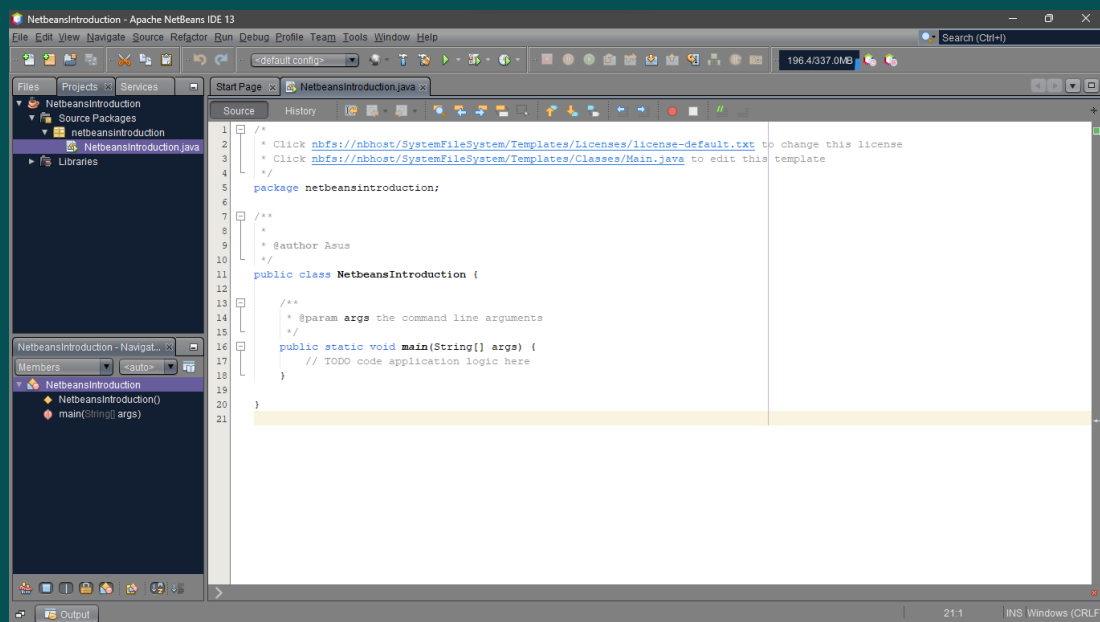
3. After that, you have to fill in your **Project Name**, choose your **Project Location** or where you want to save that, and check or uncheck the **Create Main Class** option. For example, as shown below.



Note: you must uncheck the Create Main Class option because we will create and use another class as our Main Class.

4. Click the **Finish** button after you complete all your project configurations. The project will be created and opened.

5. This is how it looks after you finish creating your project.



From the picture above, we can see more features and navigation options that you can use to build your project.

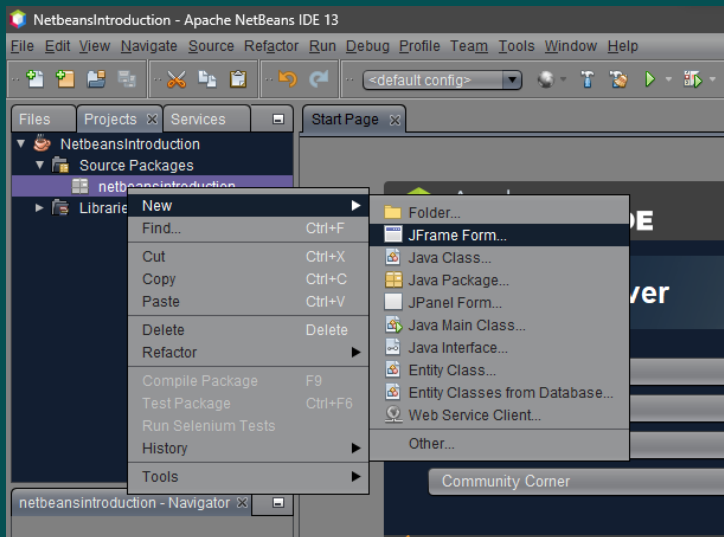
- Projects Tab: this is where you can look at all your active projects.
- Services Tab: this tab is used to configure your project with third parties like database, gradle, etc.
- Navigator Tab: This tab shows what methods are available in our active class/file.

## Creating a JFrame Container

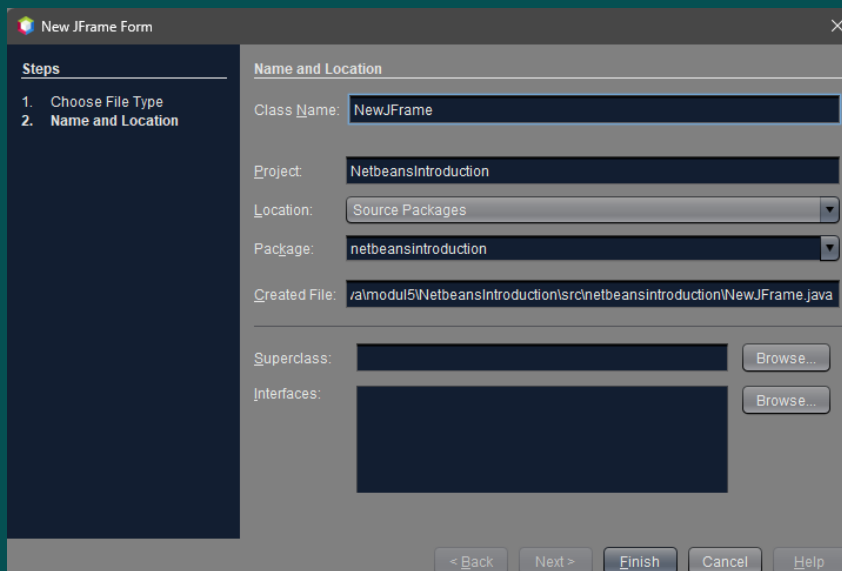
In Java GUI programming, a JFrame is a class provided by the Java Swing library that represents a window or a frame with a title bar and other components such as buttons, labels, text fields, and menus. JFrame is the top-level container that holds all the GUI components for a Java application. It provides the basic functionality of a typical windowed application

After creating the new application, you may have noticed that the Source Packages folder in the Projects window contains an empty <default package> node (if you uncheck the Create Main Class Option). To proceed with building our interface, we need to create a Java container within which we will place the other required GUI components. In this section we'll create a JFrame class and try to input or change some components inside.

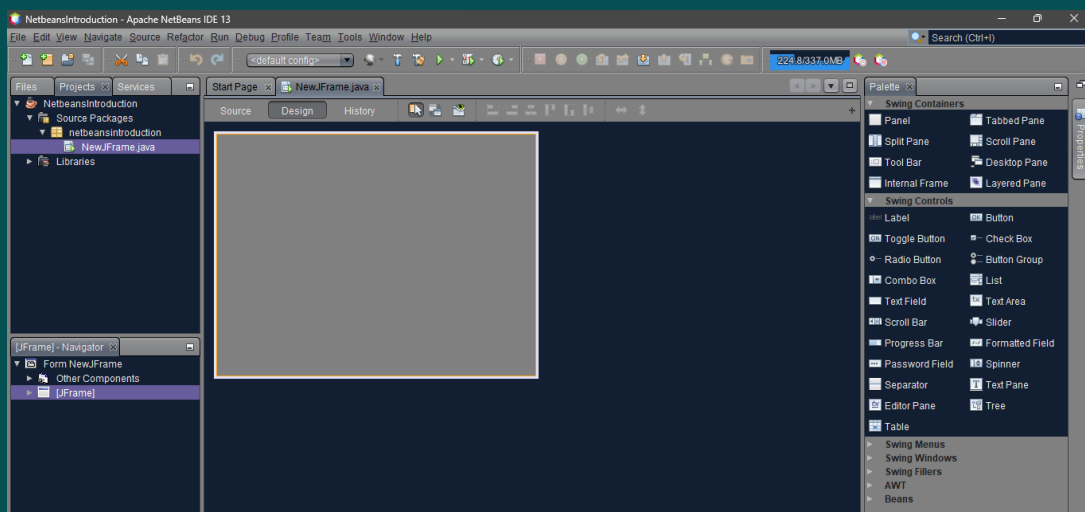
1. In the Projects window, right-click your project name and choose **New > JFrame Form**. Alternatively, you can find a JFrame form by choosing **New > Other > Swing GUI Forms > JFrame Form**.




2. There will be a dialog box to create your JForm. Fill in your JFrame class name and click Finish. When you fill in the class name, it will automatically generate your JFrame name with that.



3. This is what it looks like after you create your JForm class.



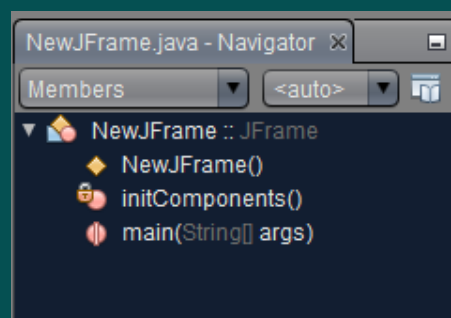


After we create the Jform, it will open in the GUI Builder's Design view and three additional windows appear automatically along the IDE's edges, enabling you to navigate, organize, and edit GUI forms as you build them. The GUI Builder's various windows include:

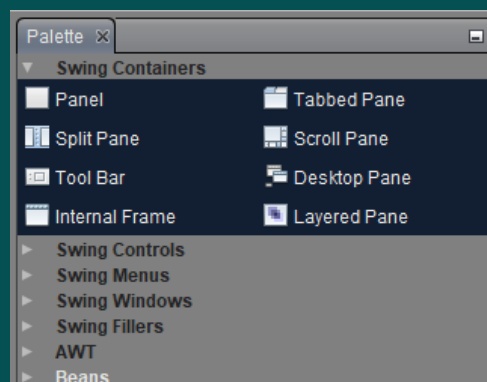
- **Design Area.** The primary window for creating and editing Java GUI forms. The toolbar's **Source button** enables you to view a class's source code, the **Design button** allows you to view a graphical view of the GUI components, and the **History button** allows you to access the local history of changes in the file. The additional toolbar buttons provide convenient access to common commands, such as choosing between Selection and Connection modes, aligning components, setting component auto-resizing behavior, and previewing forms.



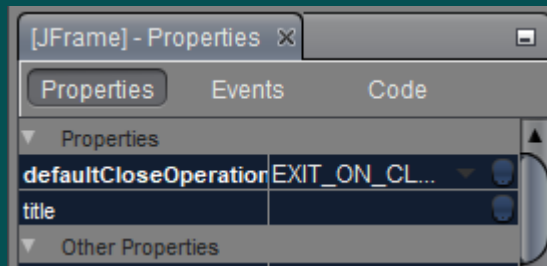
- **Navigator.** Provides a representation of all the components, both visual and non-visual, in your application as a tree hierarchy. The Navigator also provides visual feedback about what component in the tree is currently being edited in the GUI Builder as well as allows you to organize components in the available panels.



- **Palette.** A customizable list of available components containing tabs for JFC/Swing, AWT, and JavaBeans components, as well as layout managers. In addition, you can create, remove, and rearrange the categories displayed in the Palette using the customizer.

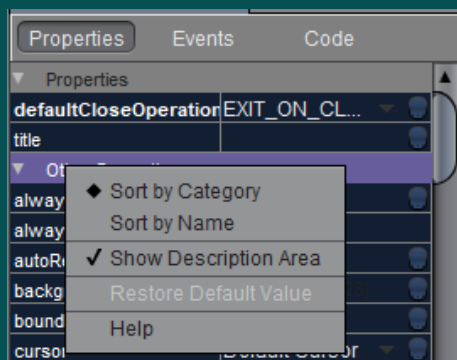


- **Properties Window.** Displays the properties of the component currently selected in the GUI Builder, Navigator window, Projects window, or Files window.

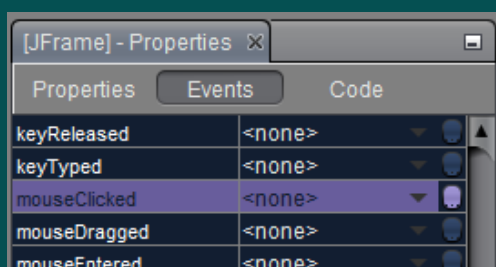


4. You can customize your component from this window which you can find beside the palette tab or you can just right click on your component and choose properties (the properties window will pop up). These properties can be modified by simply clicking on the value of the property and typing in a new value. There are three main sections within the Properties window:


- **Properties.** This section displays a list of all the properties associated with the currently selected component. The properties can be sorted by category, and each property can be expanded to show more information about it.



- **Events.** This section displays a list of all the events associated with the selected component. Events are actions that can be triggered by the component, such as a button click or a mouse movement. You can use this section to add or modify event handlers for the component. If you want to create your components can trigger an action (ex: Button can be clicked and redirect to another page), you can choose many action that you want to add (example: if you want to create your button can be clicked, you can choose mouseClicked event)



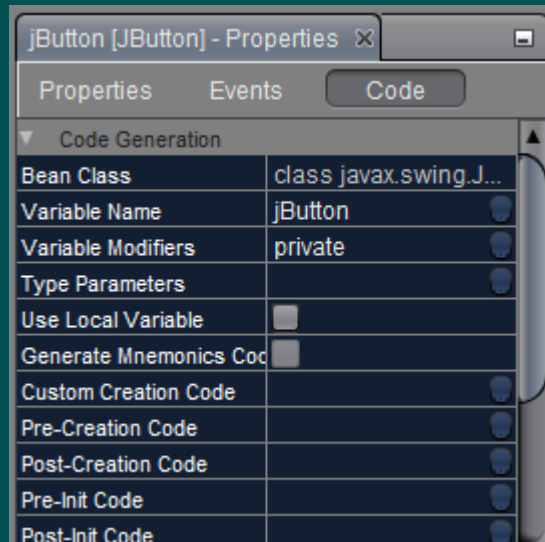




After that, you will be directed to your event method you chose before. In this method, you can insert your code and your logic so your component can run as you expected.

```
private void jButtonMouseClicked(java.awt.event.MouseEvent evt) {  
    // TODO add your handling code here:  
}
```

- **Code.** This section allows you to view and modify the code associated with the selected component. You can change your component identity (like component name, and access modifier).



## Code From Source

### Mahasiswa.java

```
public class Mahasiswa {
    private String nim;
    private String nama;
    private String nilai;

    public Mahasiswa(String nim, String nama, String nilai){
        this.nim = nim;
        this.nama = nama;
        this.nilai = nilai;
    }

    public void setNim(String nim) {
        this.nim = nim;
    }

    public void setNama(String nama) {...3 lines }

    public void setNilai(String nilai) {...3 lines }

    public String getNim() {
        return this.nim;
    }

    public String getNama() {...3 lines }

    public String getNilai() {...3 lines }
}
```

### Menu.java

### Daftar Mahasiswa

NIM

Add

Nama

Cancel

Nilai

Delete

Title 1	Title 2	Title 3	Title 4

## Menu.java - Import

```
import java.util.ArrayList;
import javax.swing.JOptionPane;
import javax.swing.table.DefaultTableModel;
```

## Menu.java - Properties

```
private DefaultTableModel dtm;
private Boolean isUpdate = false;
private int selectedID = -1;
private ArrayList<Mahasiswa> listMhs;
```

## Menu.java - Constructor

```
public Menu() {
    // Constructor
    initComponents();
    listMhs = new ArrayList<>();
    // Dummy
    listMhs.add(new Mahasiswa("2132888", "Cahya Gumilang", "A"));
    listMhs.add(new Mahasiswa("2001103", "Nelly Joy C. S.", "B"));
    listMhs.add(new Mahasiswa("2202222", "Muhammad Satria R.", "C"));
    // Set Table
    tblMhs.setModel(setTable());
    // Hide Delete button
    btnDelete.setVisible(false);
}
```

## Menu.java - setTable()

```
public final DefaultTableModel setTable() {
    // Column Title
    Object[] column = {"NIM", "Nama", "Nilai"};
    DefaultTableModel dataTabel = new DefaultTableModel(null, column);

    // Get Cell Value
    for (int i = 0; i < listMhs.size(); i++) {
        Object[] row = new Object[3]; // Array of object
        row[0] = listMhs.get(i).getNim(); // cell 0
        row[1] = listMhs.get(i).getNama(); // cell 1
        row[2] = listMhs.get(i).getNilai(); // cell 2

        dataTabel.addRow(row); // add new row
    }

    return dataTabel;
}
```

## Menu.java - insertData()

```
public void insertData() {  
    // Get Data from Form  
    String nim = fieldNim.getText();  
    String nama = fieldNama.getText();  
    String nilai = fieldNilai.getText();  
  
    // Add New Data  
    listMhs.add(new Mahasiswa(nim, nama, nilai));  
  
    // Reset Form  
  
    //Update Table  
    tblMhs.setModel(setTable());  
  
    // Show Information  
    System.out.println("Insert Success!");  
    JOptionPane.showMessageDialog(null, "Data berhasil ditambahkan!");  
}
```

## Menu.java - updateData()

```
public void updateData() {  
    // Get Data from Form  
    String nim = fieldNim.getText();  
    String nama = fieldNama.getText();  
    String nilai = fieldNilai.getText();  
  
    // set Data to Object  
    listMhs.get(selectedID).setNim(nim);  
    listMhs.get(selectedID).setNama(nama);  
    listMhs.get(selectedID).setNilai(nilai);  
  
    // Reset Form  
  
    // Update Table  
  
    // Show Information  
    System.out.println("Update Success!");  
    JOptionPane.showMessageDialog(null, "Data berhasil diubah!");  
}
```

### Menu.java - deleteData()

```
public void deleteData() {  
    // Remove Data from List  
    listMhs.remove(selectedID);  
  
    // Update Table  
  
    // Reset Form  
  
    // Show Information  
    System.out.println("Delete Success!");  
    JOptionPane.showMessageDialog(null, "Data berhasil dihapus!");  
}
```

### Menu.java - resetForm()

...

### Menu.java - action listener button Add

```
private void btnAddActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    // If Add (data not clicked)  
    if (isUpdate == false)  
        insertData();  
    else {  
        updateData();  
        btnAdd.setText("Add");  
        btnDelete.setVisible(false);  
        this.isUpdate = false;  
    }  
}
```

### Menu.java - action listener button Delete

```
private void btnDeleteActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    if (isUpdate == true) {  
        deleteData();  
        btnAdd.setText("Add");  
        btnDelete.setVisible(false);  
        this.isUpdate = false;  
    }  
}
```

## Menu.java - action listener button Cancel

```
private void btnCancelActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    // Cancel Input Form  
    btnAdd.setText("Add");  
    btnDelete.setVisible(false);  
    this.isUpdate = false;  
  
    // Reset Form  
}
```

## Menu.java - action listener table row

```
private void tblMhsMouseClicked(java.awt.event.MouseEvent evt) {  
    // TODO add your handling code here:  
    // If data clicked  
    this.isUpdate = true;  
  
    //Get Selected Data  
    int row = tblMhs.getSelectedRow();  
    String selectedNim = (tblMhs.getModel().getValueAt(row, 0).toString());  
    String selectedNama = (tblMhs.getModel().getValueAt(row, 1).toString());  
    String selectedNilai = (tblMhs.getModel().getValueAt(row, 2).toString());  
  
    // Search Data  
    for (int i = 0; i < listMhs.size(); i++) {  
        if (selectedNim.equals(listMhs.get(i).getNim())) {  
            selectedID = i;  
            break;  
        }  
    }  
  
    // Set Form Value  
    fieldNim.setText(selectedNim);  
    fieldNama.setText(selectedNama);  
    fieldNilai.setText(selectedNilai);  
  
    btnAdd.setText("Update");  
    btnDelete.setVisible(true);  
}
```

## Exercise

Download this starter project: [starter project](#)

- Add more property (component type: any, except text field)
- Add method to reset form
- Refresh table after update and delete
- Add confirmation prompt before delete
- Build project (generate .jar file)

Result:

NIM	Nama	Nilai	Gender
2132888	Cahya Gumil...	A	Laki-laki
2001103	Nelly Joy C. S.	B	Perempuan
2202222	Muhammad ...	C	Laki-laki

Note.

- Program dikumpulkan pada repository GitHub yang dibuat public dengan nama “LATIHAN5DPBO2023”
  - Hanya program pada branch **main** yang akan dinilai dan diperiksa
  - Jika waktu pengumpulan sudah habis dan ingin mengupdate kode program, update pada branch lain karena mengupdate branch Main setelah waktu pengumpulan terlewat maka program tidak akan dinilai
- Struktur folder
  - source\_code
  - program
  - + Screenshot
  - + JAR



## README.md

- File README berisi desain program, penjelasan alur, dan dokumentasi saat program dijalankan (screenshot/screen record)
- Submit link repository pada form berikut:  
<https://forms.gle/rvb1hKxbQVuYNbhKA>





## Penutup

Terima kasih atas kerja sama seluruh pihak yang membantu dalam penyusunan modul ini, semoga apa yang telah didapatkan bisa bermanfaat di masa yang akan datang.

## Daftar Pustaka

Sukamto, Rosa A. (2018). *Pemrograman Berorientasi Objek*. Bandung: Modula.

Asisten Pemrograman 11. (2022). *Modul Desain dan Pemrograman Berorientasi Objek*.