



# Chapter 3:

## Getting Started with JAVA IDE Programming

**Informatics Practices**  
Class XI (CBSE Board)

Revised as per  
CBSE  
Curriculum  
2015

**"Open Teaching-Learning Material"**

Visit [www.ip4you.blogspot.com](http://www.ip4you.blogspot.com) for more....

**Authored By:-** Rajesh Kumar Mishra, PGT (Comp.Sc.)  
Kendriya Vidyalaya Upper Camp, Dehradun (Uttarakhand)  
e-mail : rkmalld@gmail.com

# Learning Objectives

---

In this presentation, you will learn about-

- ❑ Introduction to JAVA
  - ❑ History of JAVA
  - ❑ Characteristics of JAVA
  - ❑ Basics of GUI Applications
  - ❑ Introduction to NetBeans IDE
  - ❑ Developing Simple Applications with Netbeans
  - ❑ Working with Java Swing Controls-Buttons, TextFields, Labels etc.
  - ❑ Creating Even-Handling Methods
-

# What is JAVA?

---



- ❑ JAVA is an Object Oriented programming language as well a platform.
  - ❑ By using JAVA, we can write Platform independent application programs, which can run on any type of OS and Hardware.
  - ❑ JAVA is designed to build Interactive, Dynamic and Secure applications on network computer system.
  - ❑ Java facilitates development of Multilingual applications because JAVA uses 2-Byte UNICODE character set, which supports almost all characters in almost all languages like English, Chinese, Arabic etc.
-

# History of JAVA

---

JAVA was developed by **James Gosling** at **Sun Microsystems** under the **Green project** to write applications for electronic devices like TV-Set Top Box etc. The language was initially called **Oak** and later renamed with **Java**.



James Gosling

1991	James Gosling developed Oak to program consumer electronic devices
1995	Java Development Kit (JDK) 1.0 was released by the Sun Microsystems and JAVA used as a part of Netscape web browser to facilitate Internet Applications.
1998	Sun introduced "Open" source community and produces JDK 1.2 (Java 2) which was released as J2EE, J2SE, J2ME version.
2006	Sun declared Java as Free & Open Source Software (FOSS) under GNU-GPL and NetBeans IDE was released.
2010	Sun Microsystems was owned by Oracle Corporation and now Java Project is being governed by the Oracle.

---

# Characteristics of JAVA

---

## ❑ Object Oriented Language

Java is Object Oriented Language (a real-world programming style)

## ❑ Open Source Product

It is Open Source i.e. freely available to all with no cost.

## ❑ Write Once Run Anywhere (WORA)

JAVA Program can be run on any type of H/W and OS platforms i.e. Java programs are platform independent.

## ❑ Light Weight Code

Big applications can be developed with small code.

## ❑ Security

JAVA Programs are safe and secure on Network.

## ❑ Interpreter & Compiler based Language

JAVA uses both Compiler and Interpreter (JVM) to produce portable and platform-independent object code.

## ❑ Built-in Graphics & Supports Multimedia

JAVA is equipped with Graphics feature. It is best for integration of Audio, Video and graphics & animation.

---



# Why JAVA is Platform Independent?

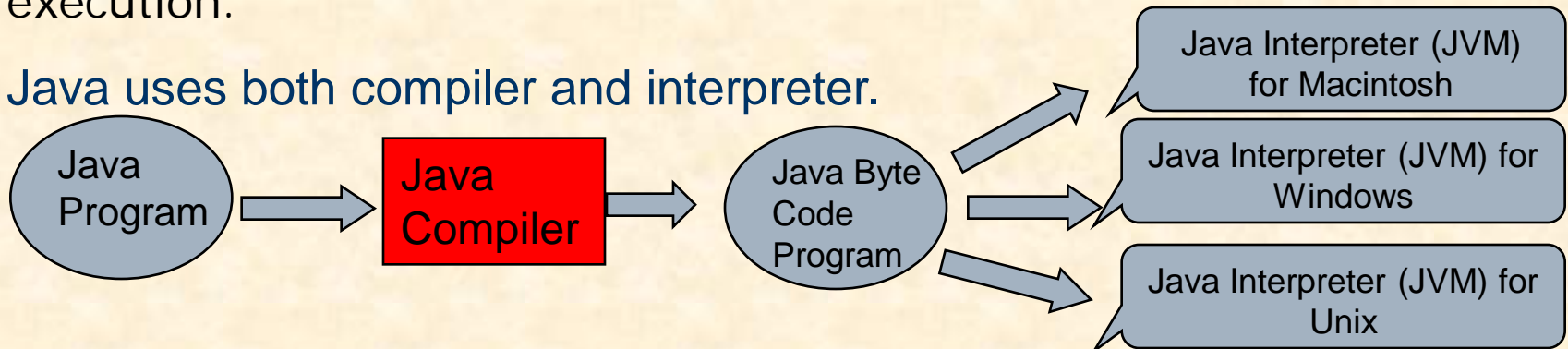
---

A program written in HLL must be converted into its equivalent Machine code, so that computer can understand and execute. This conversion is known as Compilation. Generally, the converted machine code depends on the structure of H/w and OS platform. So that a Windows program will not work on UNIX, or LINUX or Mac platform etc. Since they are Platform dependent.

A program written in JAVA is platform-independent i.e. they are not affected with changing of OS. This magic is done by using Byte code. Byte code is independent of the computer system it has to run upon.

Java compiler produces **Byte code** instead of native executable code which is interpreted by Java Virtual Machine (**JVM**) at the time of execution.

Java uses both compiler and interpreter.



# Basics of GUI Applications

---

## □ How GUI application works ?

Graphical User Interface (GUI) based application contains Windows, Buttons, Text boxes, Dialogue boxes and Menus etc. known as GUI components. While using a GUI application, when user performs an action, an **Event** is generated which causes a **Message** sent to application to take action.

## □ What is Event ?

An Events refers to the occurrence of an activity by user like click, double click etc.

## □ What is Message ?

A Message is the information/request sent to the application when event occurs.

---

# GUI in JAVA

---

- ❑ In JAVA, the GUI programming is done through **Swing Controls** which are available as a part of Java Foundation Classes (JFC).
  - ❑ A GUI application in JAVA contains **three** basic elements.-
    - 1. Graphical Component (Event Source):**

It is an object that defines a screen element such as Button, Text field, Menus etc. They are source of the Events. In GUI terminology, they are also known as **Widget** (Window Gadget). It can be **container** control or **child** control.
    - 2. Event:**

An Event (occurrence of an activity) is generated when user does something like mouse click, dragging, pressing a key on the keyboard etc. Events are trapped by the Event Listeners.
    - 3. Event Handler Method:**

It contains method/functions which is attached to a component and executed in response to an event. Generally, *ActionPerformed* Event-Handler method is used.
-



# Events Handling in JAVA GUI Application

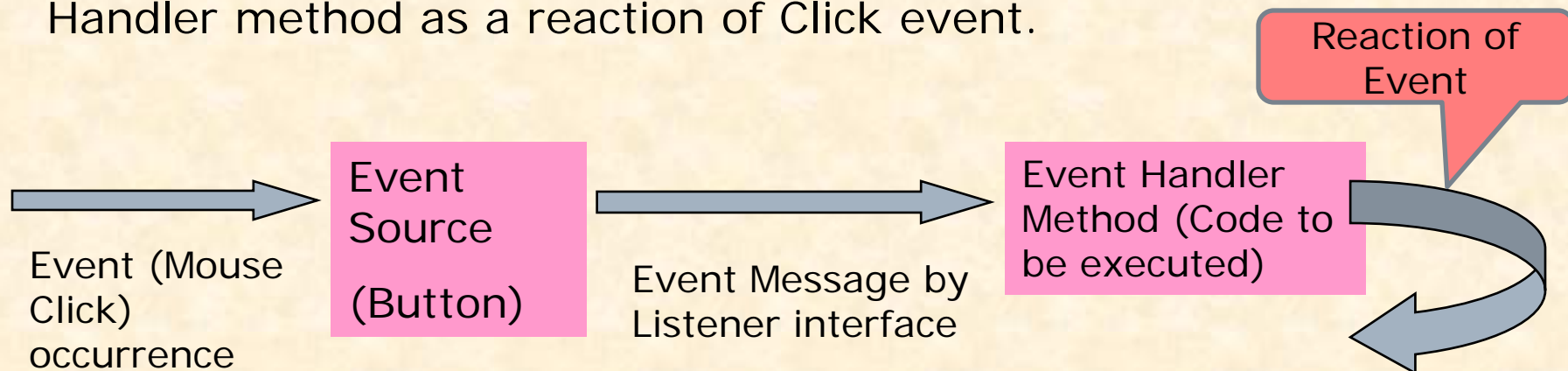
---

JAVA GUI Event Handling System is also based on Event, Listener Interface and Event-Handler Method.

In a GUI Applications, a user generates an events like pressing keys, or Mouse Click etc. on GUI controls like text fields or buttons etc.

When an event occurs, it is trapped by **Listener Interface** (Keyboard Listener, Mouse Listener etc.) and **Event Message** is passed to **Event-handler** Method for action. Event-handler methods contains code to be executed in response to an event.

For example, if you press a Button, the Mouse Event is generated and trapped by Mouse listener. Listener Interface executes Event-Handler method as a reaction of Click event.



# Basic GUI Controls of JAVA(Swing controls)

---

The Java offers various Swing controls to facilitate development of GUI applications. Most commonly used controls are-

- **JFrame**: Used as a Basic Window or form.
- **JLabel**: Allows Non-editable text or icon to be displayed.
- **TextField**: Allows user input. It is editable through text box.
- **Button**: An action is generated when pushed.
- **CheckBox**: Allow user to select multiple choices.
- **RadioButton**: They are option button which can be turned on or off. These are suitable for single selection.
- **JList**: Gives a list of items from which user can select one or more items.
- **ComboBox**: It gives dropdown list of items with facility to add new items. It is combination of JList and TextField.
- **JPanel**: It is container controls which contains other controls using a frame.

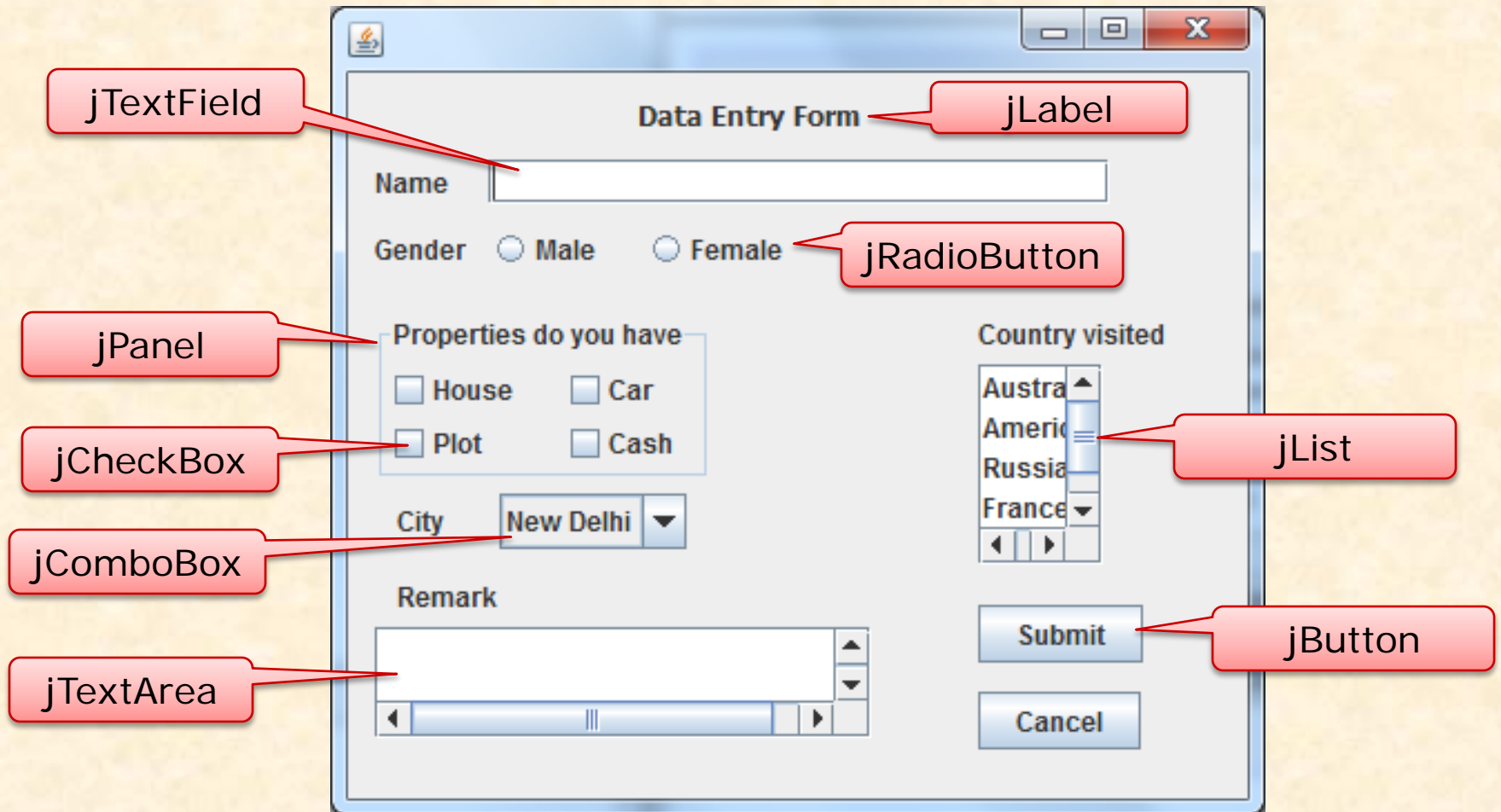


A control which holds other component (child) controls, is called **Container Control** e.g. JFrame, JPanel, JDialog etc.

---

# GUI Controls (Swing Controls) in JAVA

---



# Introduction to JAVA's IDE - NetBeans

---

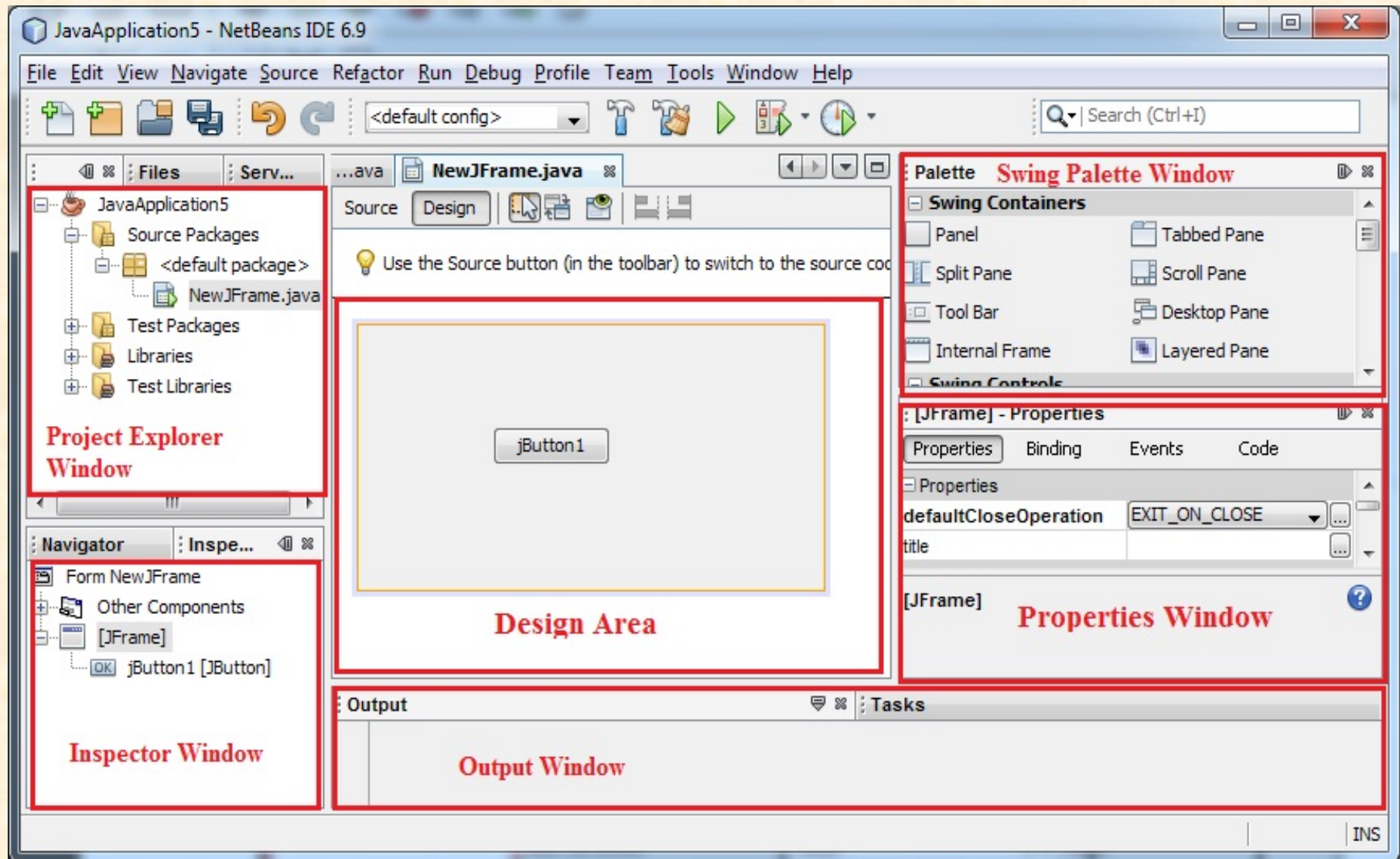
- ❑ NetBeans IDE is a free, open-source, cross platform Integrated Development Environment (IDE) which supports Java programming. It is equipped with the most advanced GUI building tools to facilitate **Rapid Application Development (RAD)**.
- ❑ NetBeans offers the following features-
  1. Drag & Drop GUI creation.
  2. Advanced Source Code Editor.
  3. Excellent Debugging tools.
  4. Wizards & Code generator
  5. Project Management tools.

## What is RAD ?

RAD describes a methodology of software development through the use of pre-programmed tools or wizards. The programmed tools or controls are simply dragged & dropped on a screen to visually design the interface of application so Application can be developed faster.

Example of such RAD IDEs are- VB IDE, Eclipse, NetBeans etc.

# Working with NetBeans IDE





# Components of NetBeans IDE

---

## Title Bar:

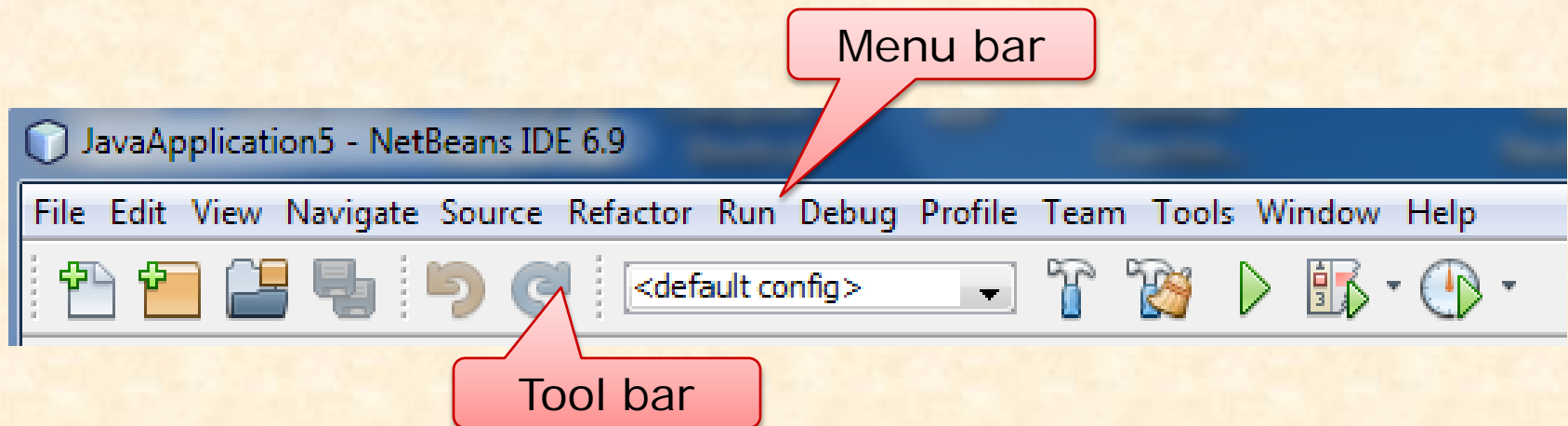
The topmost line of the Netbeans IDE Window is called Title Bar, which contains title of the opened project.

## Menu Bar & Pull Down Menus:

Menu Bar are group of Menus which are opened as Pull Down, when clicked. These menu offers choices of action.

## Toolbar :

These are pictorial buttons which shorthand representations of Menus options.



# Components of NetBeans IDE

---

## **Design Area (GUI Builder):**

It is used for creating and editing Java GUI forms. You can also preview the designed form.

## **Inspector Window:**

Provides a graphic representation of all the components attached in our application as a tree hierarchy. It also provides details of the component currently being edited in the GUI Builder.

## **Swing Palette Window:**

Contains a list of components containing tabs for Swing controls, and JavaBeans components.

## **Properties Window:**

Displays the properties of the component currently selected in the GUI Builder, Inspector window or Projects window.

## **Project Explorer Window:**

Displays the information about currently and recently opened Projects.

## **Output Window:**

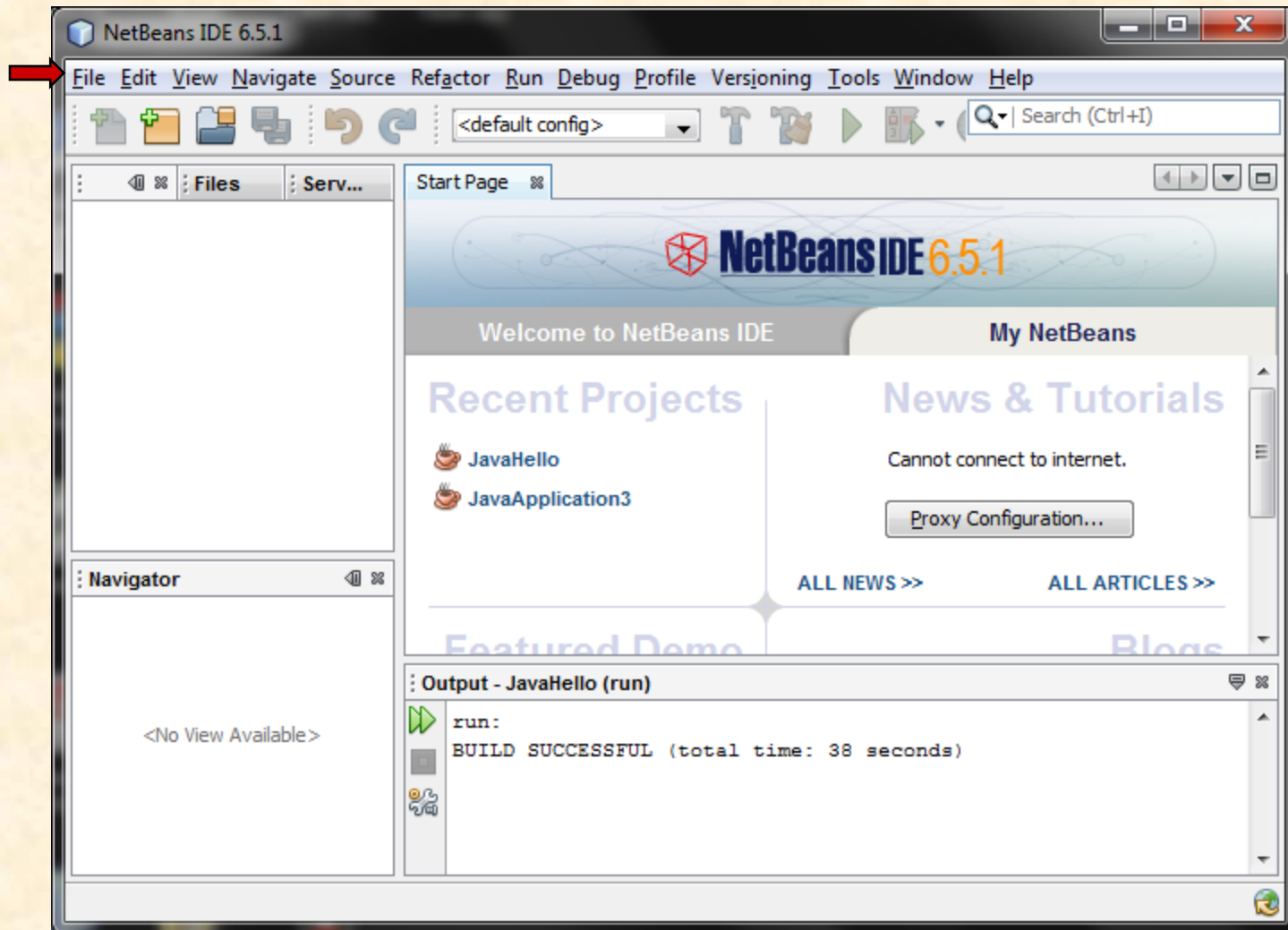
Output Window displays the output as well as Error description while writing/compiling of the application.

---

# How to create a JAVA Application

---

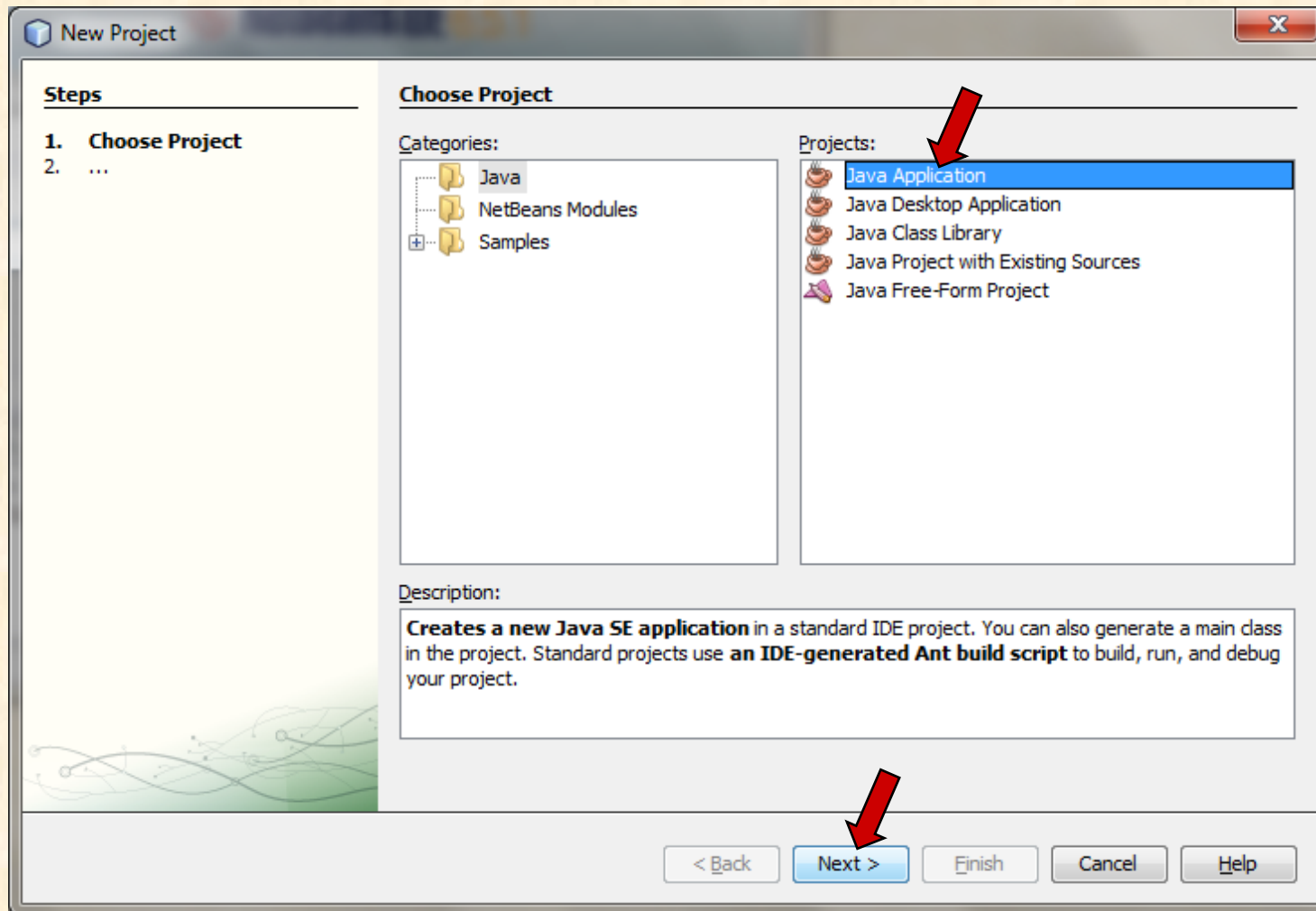
Start NetBean and choose **File->New Project**



# How to create a JAVA Application

---

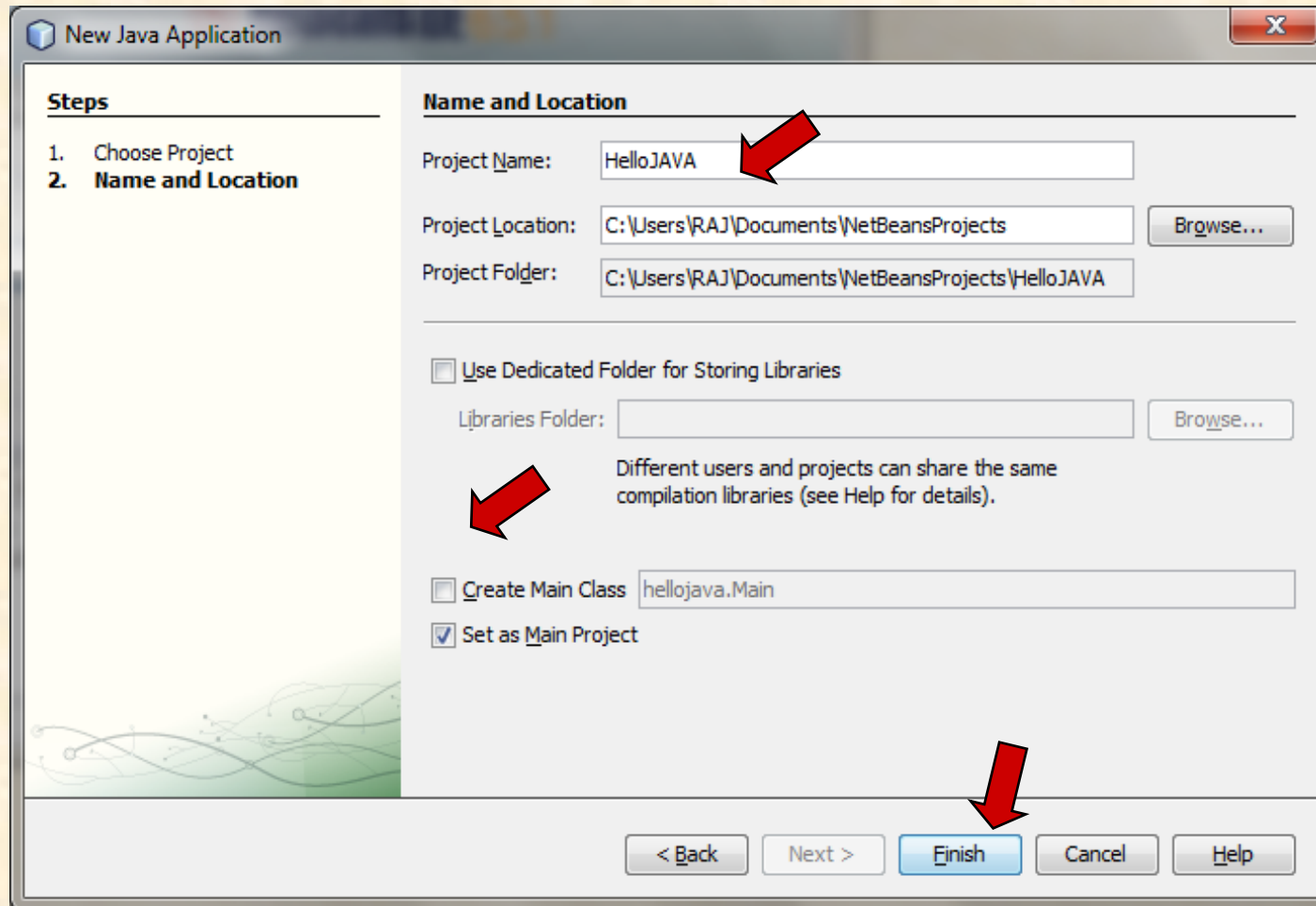
Select **Java Application** and Press **Next** button



# How to create a JAVA Application

---

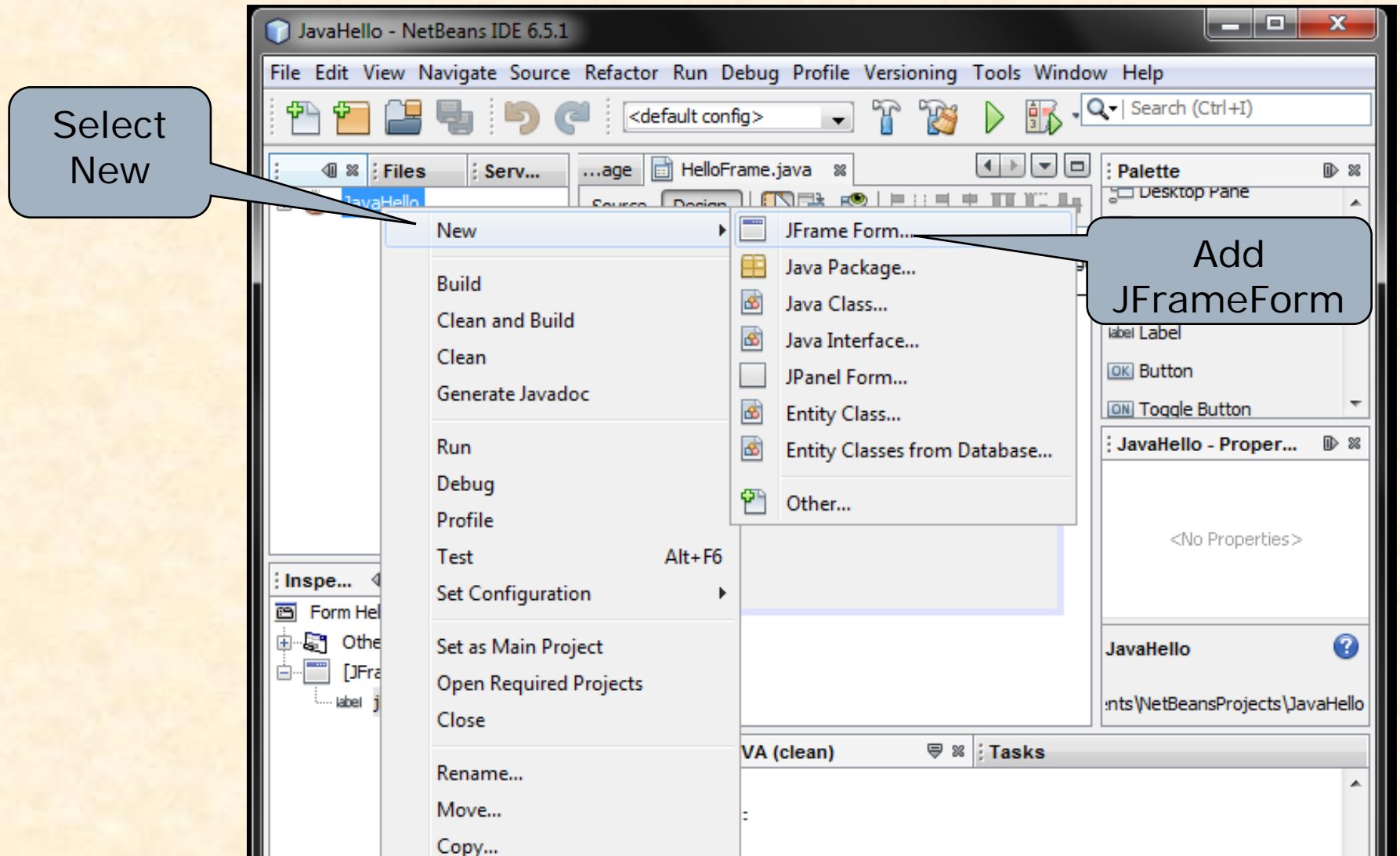
Give Project Name ex. *HelloJAVA*. Uncheck create Main Class and Press **Finish** Button





# How to create a JAVA Application

---



# How to create a JAVA Application

---

Give a Class Name and Press **Finish** Button

**New JFrame Form**

**Steps**

1. Choose File Type
2. **Name and Location**

**Name and Location**

Class Name: HelloFrame

Project: HelloJAVA

Location: Source Packages

Package:

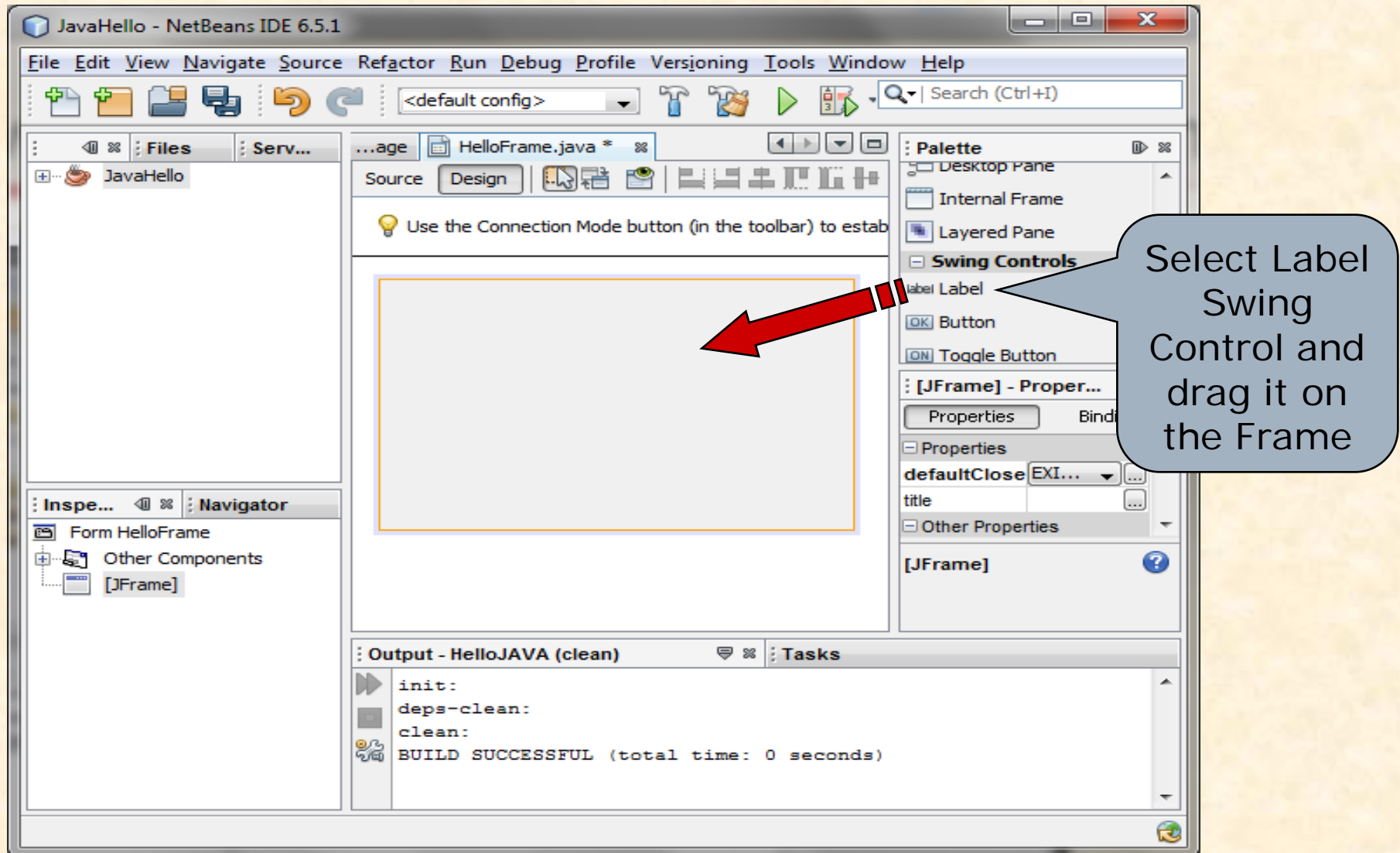
Created File: C:\Users\RAJ\Documents\NetBeansProjects\HelloJAVA\src\HelloFrame.java

⚠ Warning: It is highly recommended that you do NOT place 1 classes in the default pack...

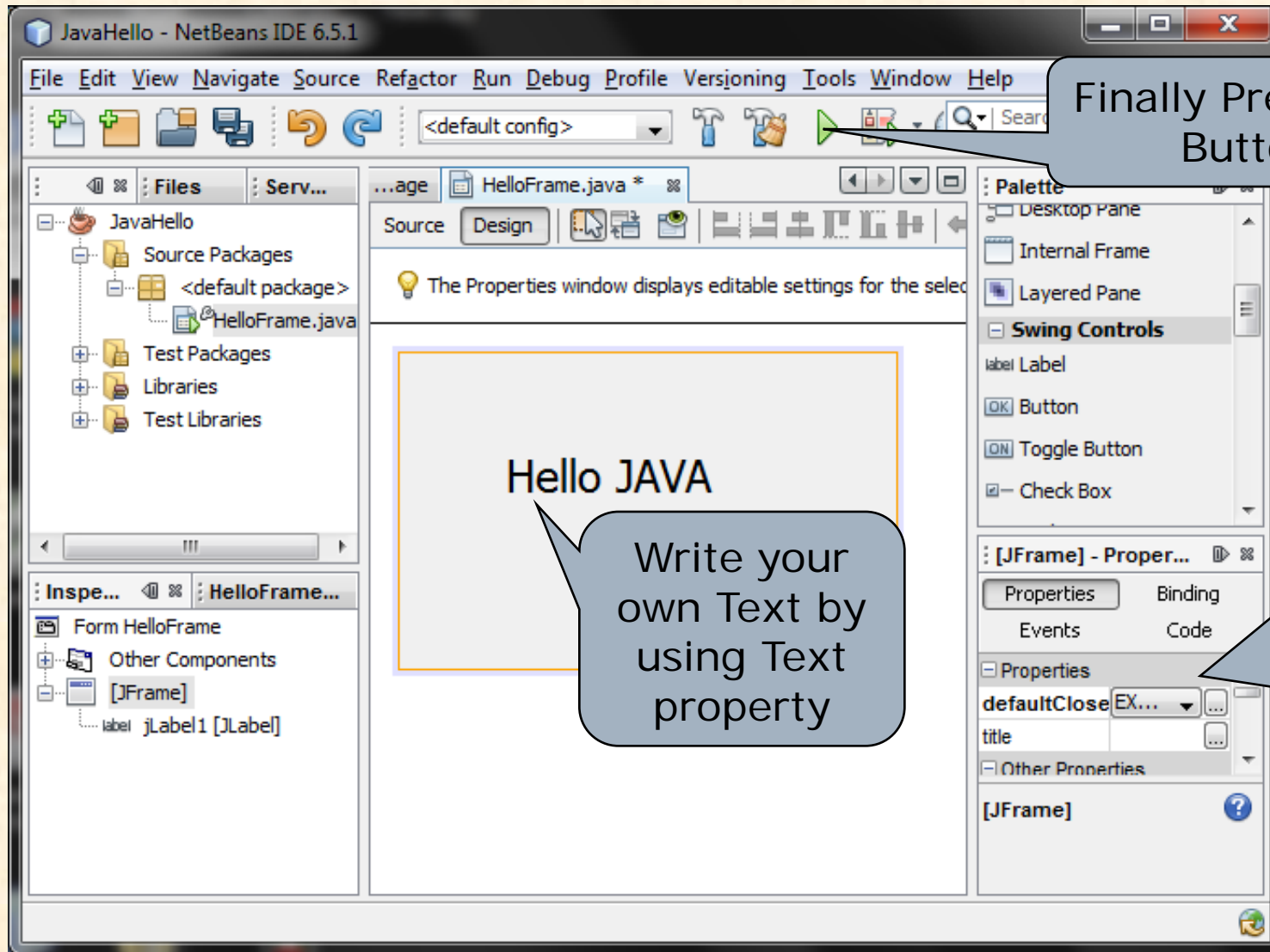
< Back   Next >   **Finish**   Cancel   Help

# How to create a JAVA Application

---



# How to create a JAVA Application



Finally Press Run Button

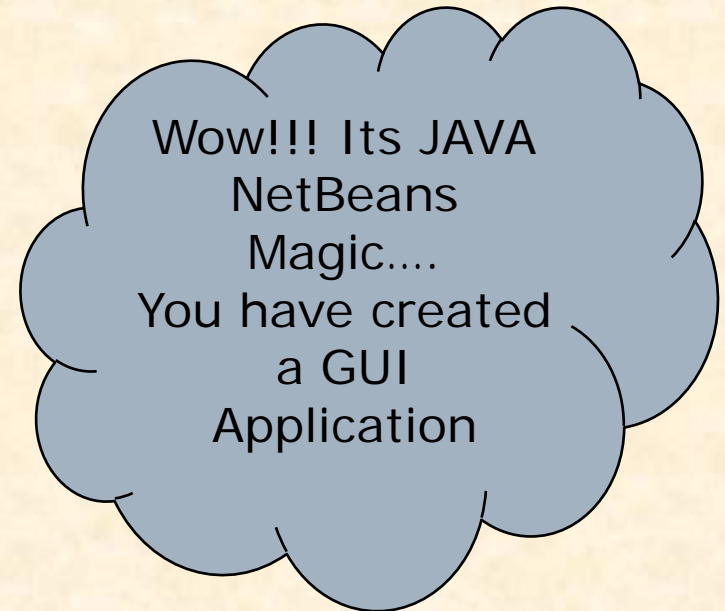
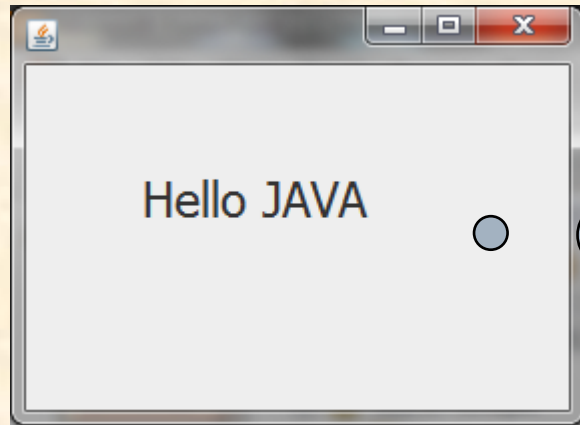
Hello JAVA

Write your own Text by using Text property

Customize Label control through Text & Font Property

# How to create a JAVA Application

---





# Customizing Control's Properties

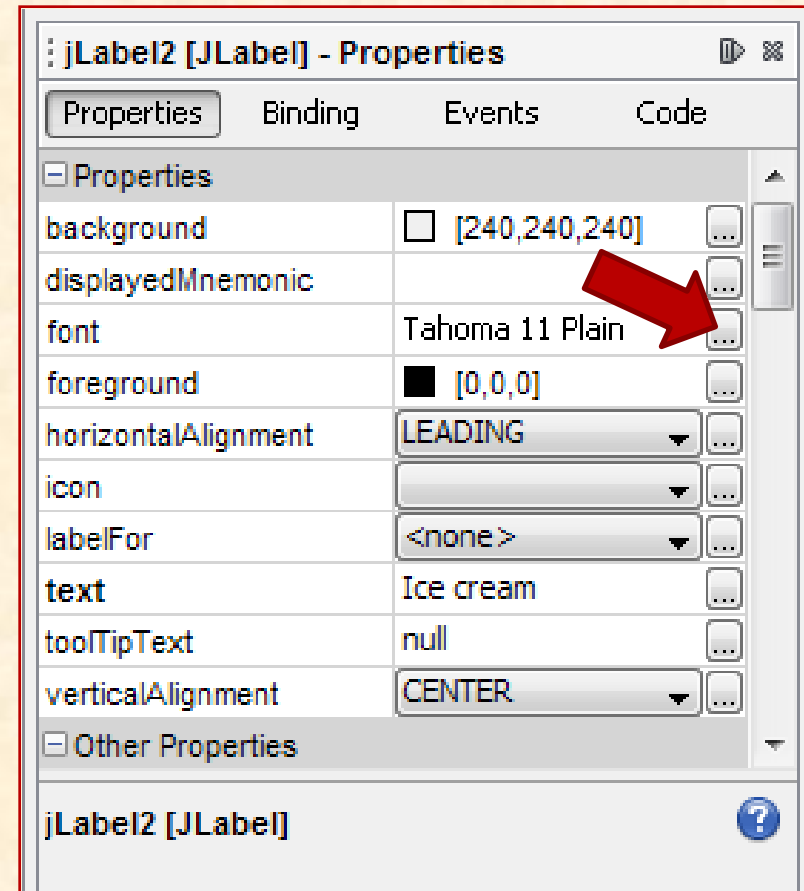
A Swing control like Label, TextField, Button etc. can be customized as per need using Properties Box.

To change various properties-

1. Select control
2. Click on (...) (ellipse button) of desired property and set/change values.

Commonly used properties are-

Text, Font, Background,  
Foreground, ToolTipText,  
MinimumSize, Height,  
MaximumSize, Width and Icon etc.



Properties are used to control the appearance of a control on the Form.

# Using Button control

---

A Button control commonly used to make an action when clicked by user. Generally an Event-handler method is attached with Button control, which comprises a set of JAVA commands to be executed to perform an action.

To attach a Button control on the Frame-

- Select **Button** component from Swing Palette and drag on the frame on desired place. It will create *jButton1* object.
- By default a button control bears 'jButton1' text on it. To write your own text (caption) on button, just Right click on the *jButton* control and select **Edit Text** option and write your own text. (Alternatively you may use *Text* property)
- Resize the button control, if required.
- You may also set **Font**, **size**, **Background** and **Foreground** color etc. properties from Properties Window, if required.

Once *jButton* control has been placed on the form, you may attach code as Event-Handler method to perform an action.

---

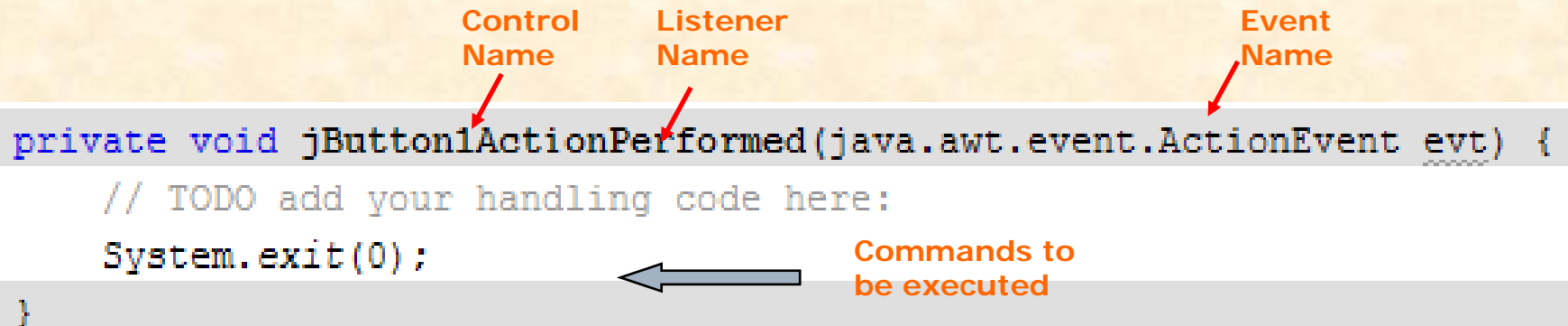
# Attaching Code with Button Control

---

To perform some action by the button, you have to attach a set of JAVA commands to be executed in Event-Handler Method. Generally, `ActionPerformed` Event-Handler method is attached with Button control.

To attach an event-handler method with a Button control-

- Double Click on the Button (JButton1) Control, it will open source Code editor with default Handler-Name, where you can type TO DO code for the Event-Handler method. (Alternatively, Right Click on the button and select- *Events* > *Action* > *ActionPerformed* event)



The diagram shows the following code snippet with annotations:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    System.exit(0);  
}
```

Annotations:

- Control Name**: Points to `jButton1` in the method signature.
- Listener Name**: Points to `ActionPerformed` in the method signature.
- Event Name**: Points to `ActionPerformed` in the parameter list.
- Commands to be executed**: Points to the body of the method, specifically the `System.exit(0);` line.

# Using Message Dialog Box (JOptionPane)

---

Java offers ready to use Message DialogBox to display any information or message to user. Message dialog box belongs to JOptionPane class available in Swing Library.

To use the JOptionPane dialog in the application, you must import the JOptionPane class (at top of the code window) as-

```
import javax.swing.JOptionPane;
```

In general, the following syntax of methods along with optional parameters can be used-

```
JOptionPane.showMessageDialog([frame name], <"Message"> );
```

❑ **Frame Name:** Generally **null** is used to indicate current frame.

❑ **Message:** User given string to convey the message.

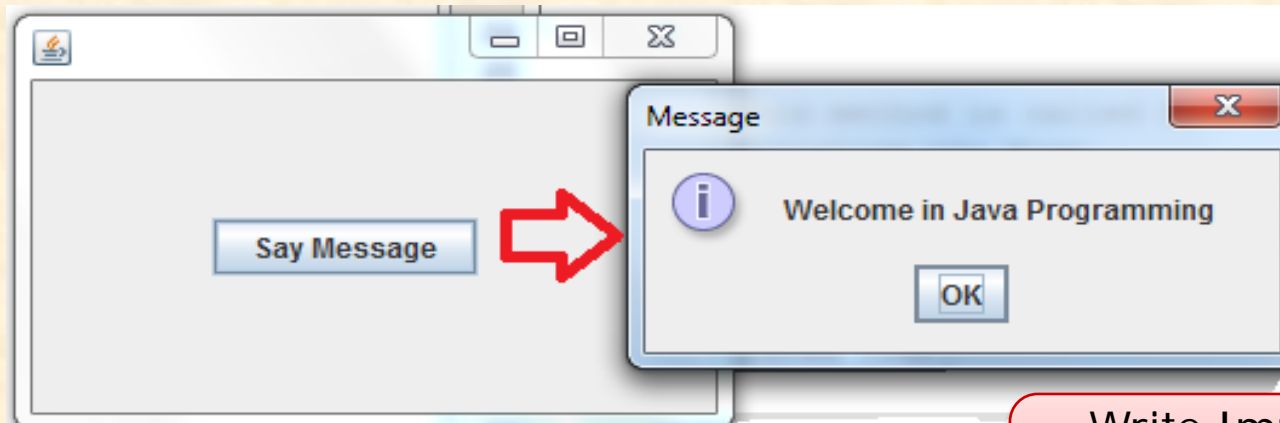
**Example:**

```
JOptionPane.showMessageDialog(null, "JAVA Welcomes You");
```

```
JOptionPane.showMessageDialog (null, "My Name is "+jTextField1.getText());
```

---

# Demo-Message DialogBox



Message  
Dialog  
Box

Write Import command for  
JOptionPane library at the top  
of the code window

```
1
2 import javax.swing.JOptionPane;
3
4 /*
5  * To change this template, choose Tools | Template
6  * and open the template in the editor.
```

Command to display  
message

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:
    JOptionPane.showMessageDialog(null, "Welcome in Java Programming");
}
```



# Using TextField control

---

A TextField control used to **get input** from user or to **display text**. It allows user to enter a **single line of text**.

To attach a TextField control on the Frame-

- Select TextField component from Swing Palette and drag on the frame on desired place. It will create *jTextField1* object.
- Double click on TextField control and erase text.
- Resize Text box, if required.
- You may set **Font,size, Background** and **Foreground** color etc. properties from Properties Window, if required.

Once *jTextField1* control has been attached, you may use the following methods in Event Handler Method (TODO Code)-

❖ To display text message-

```
jTextField1.setText("Message");
```

❖ To read input from user-

```
jTextField1.getText();
```

---

# Using TextField control

---

## Example : How to use TextField control-

```
jTextField1.setText("Hello.. Java");
```

❑ You can add two string with .setText() using + operator.

```
jTextField1.setText("Hello.." + "Java");
```

❑ You can add two string using .concat() method.

```
jTextField1.setText("Hello..".concat("Java"));
```

❑ An user input is always in string form.

```
jTextField2.setText("Hello.." + jTextField1.getText());
```

```
jTextField2.setText("Hello..".concat(jTextField1.getText()));
```

❑ A number should be converted into string form.

```
jTextField1.setText("Rs." + Integer.toString(500));
```

❑ Java automatically convert number is string when added with string.

```
jTextField1.setText("Rs." + 500);
```

```
jTextField2.setText("" + (10 * Integer.parseInt(jTextField1.getText()));
```

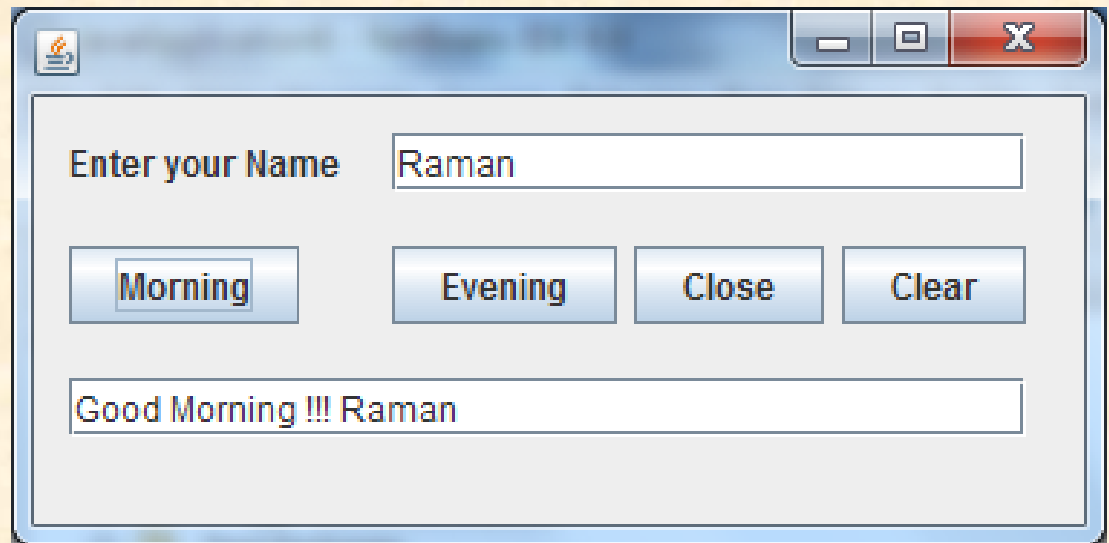
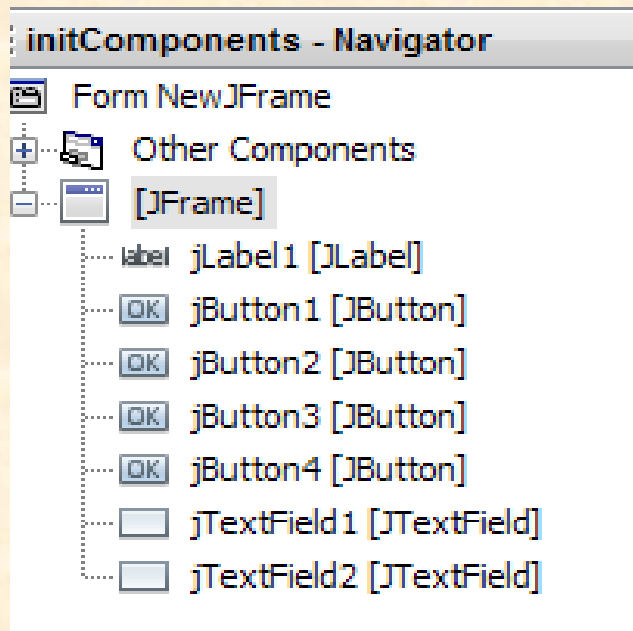
---

# Demo- TextFields and Buttons

---

Design an application to wish you Good Morning or Good Evening along with the name given by user. The controls to be used –

- jTextField1 – to accept name of the person
- jTextField2 - to display wish message
- jButton1 & 2 - to say Good Morning/Evening wish
- JButton3 - To close the application
- JButton4 - To clear the text Boxes.



# Demo- TextFields and Buttons

---

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    jTextField2.setText("Good Morning !!! " + jTextField1.getText());  
}  
  
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    jTextField2.setText("Good Evening !!! ".concat(jTextField1.getText()));  
}  
  
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    System.exit(0);  
}  
  
private void jButton4ActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    jTextField1.setText("");  
    jTextField2.setText("");  
}
```

You can also use concat() method instead of + to add strings

# Renaming Controls

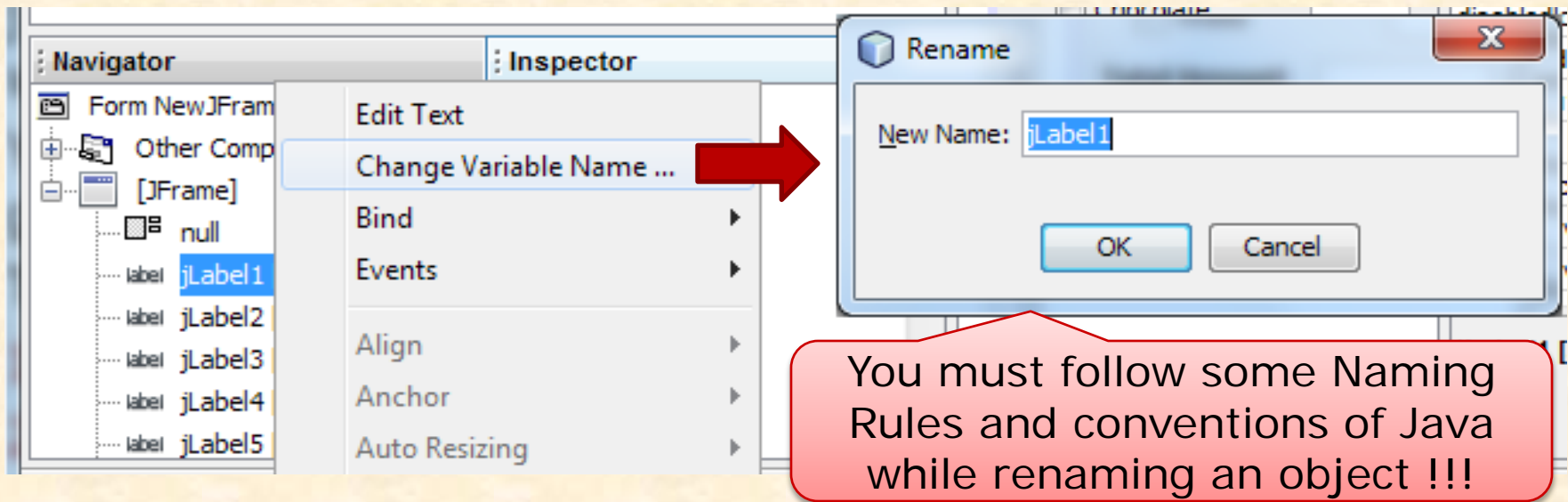
---

A Swing control bears its default name like JLabel1, JTextField1, JTextField2 and JButton1 etc., when attached with a frame.

You can give a meaningful name to a control for easy reference.

To change the default name of a Swing control -

1. Select control's node in Inspector Window.
2. Right Click and choose 'Change Variable Name' option.
3. Type new name in Rename Dialog Box and press OK.



# Object Naming Rules and Conventions

---

A Swing control object in Java, given a default name like `jTextField1`, `jButton1` etc. as they drawn on the form. But we can rename them and can give user friendly name to them for easy reference as per the following Rule and conventions-

- ❑ Name must begin with a letter.
- ❑ Name must contain only letters, numbers, and `_` .
- ❑ Punctuations (`,` `;` `.`), other symbols (`+`, `*` `@` etc.) and spaces are not allowed.
- ❑ The letter 'j' may be omitted to prefix some user familiar name or text like *Txt* (Text Fields), *Btn* (Button), *Lbl* (Label) etc. which can be used to identify a control.
- ❑ When an user familiar name is given, first letter is generally capitalized.

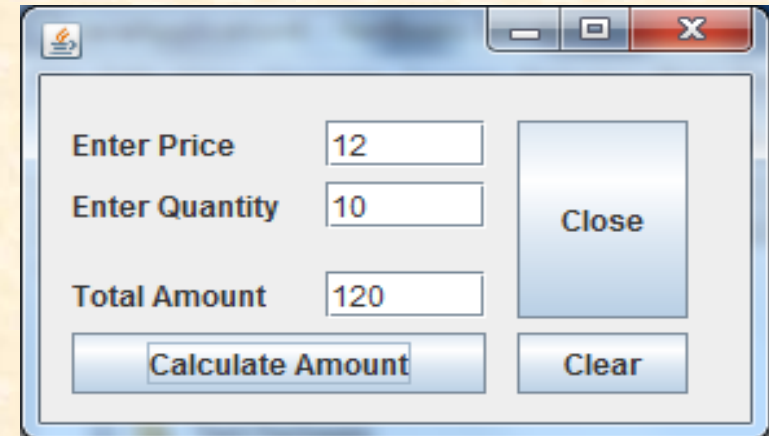
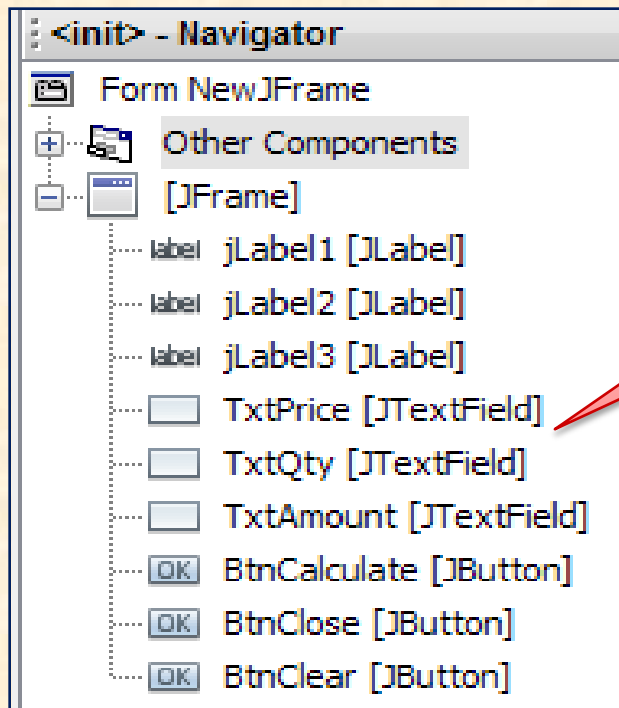
**Example-** `jButton1` (for OK) can rename as **BtnOK** and `jTextField1` to enter Amount can be renamed as **TxtAmount**.

---



# Java Application – Using Renamed controls.

Consider the following Amount calculator Application-



```
private void BtnClearActionPerformed(java  
    // TODO add your handling code here:  
    TxtPrice.setText("");  
    TxtAmount.setText("");  
    TxtQty.setText("");  
}
```

```
private void BtnCalculateActionPerformed(java.awt.event.ActionEvent evt) {  
    // TODO add your handling code here:  
    TxtAmount.setText(""+(Integer.parseInt(TxtPrice.getText())  
        *Integer.parseInt(TxtQty.getText())));  
}
```