EELU
الجامعة المصرية للتعلم الإلكتروني
Egyptian E-Learning University

SWE203

Programming Techniques (3)

GUI

(Graphical User Interface)

Part I

Previously

- Exceptions ✓
- The Random class ✓
- Files I/O ✓
- Random access files ✓
- GUI

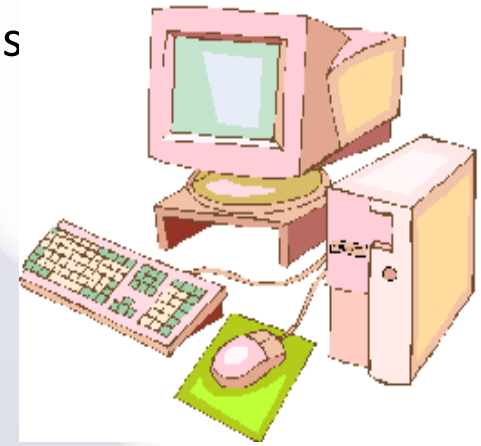
GUIs—Graphical User Interfaces

- In a text-based UI the commands are entered from the keyboard.
- In a console program, the system usually controls user actions

  > Enter number of classes: 3

  > Enter number of students: 15

  > You have 45 students



- Most modern programs use a GUI (pronounced "gooey"):
- **G**raphical: Not just text or characters but windows, menus, buttons, ..
- **U**ser: Person using the program
- **I**nterface: Way to interact with the program

<u>Graphical elements include</u>:

- Window: Portion of screen that serves as a smaller screen within the screen
- Menu: List of alternatives offered to user
- Button: Looks like a button that can be pressed
- Text fields: The user can write something in

A New Approach to Programming

Previous Style of Programming:

- List of instructions performed in order
- Next thing to happen is next thing in list
- Program performed by one agent—the computer

Event-Driven Style of Programming:

- Objects that can fire events and objects that react to events
- Next thing to happen depends on next event
- Program is interaction between user and computer

Event-Driven Programming

- Programs with **GUIs** often use Event-Driven Programming
- A user interacts with the application by:
  - Clicking on a button to choose a program option.
  - Making a choice from a menu.
  - Entering text in a text field.
  - Dragging a scroll bar.
  - Clicking on a window's close button.
- Program waits for events to occur and then responds
- Firing an event: When an object generates an event
- Listener: Object that waits for events to occur
- Event handler: Method that responds to an event

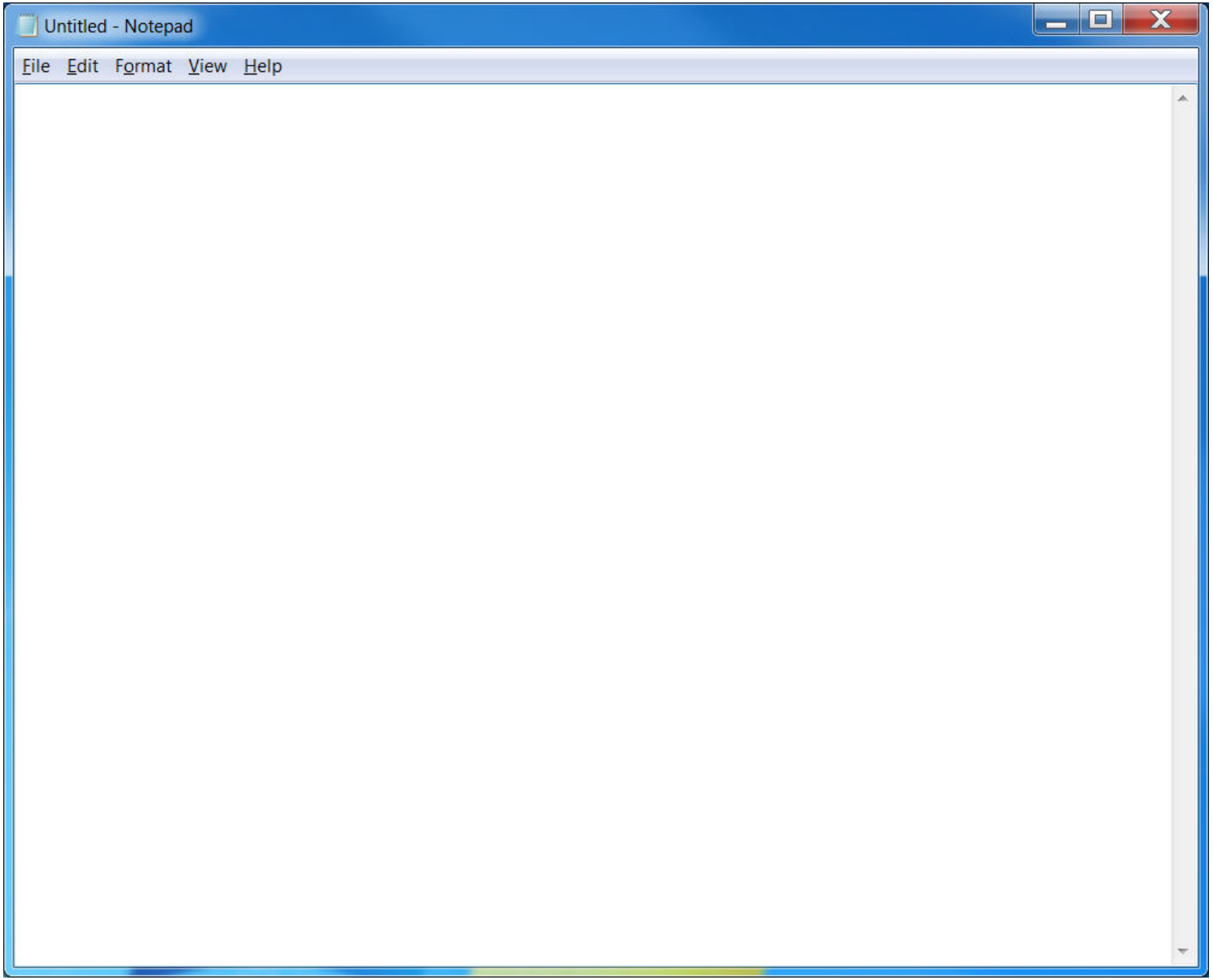
GUI

- Graphical User Interface

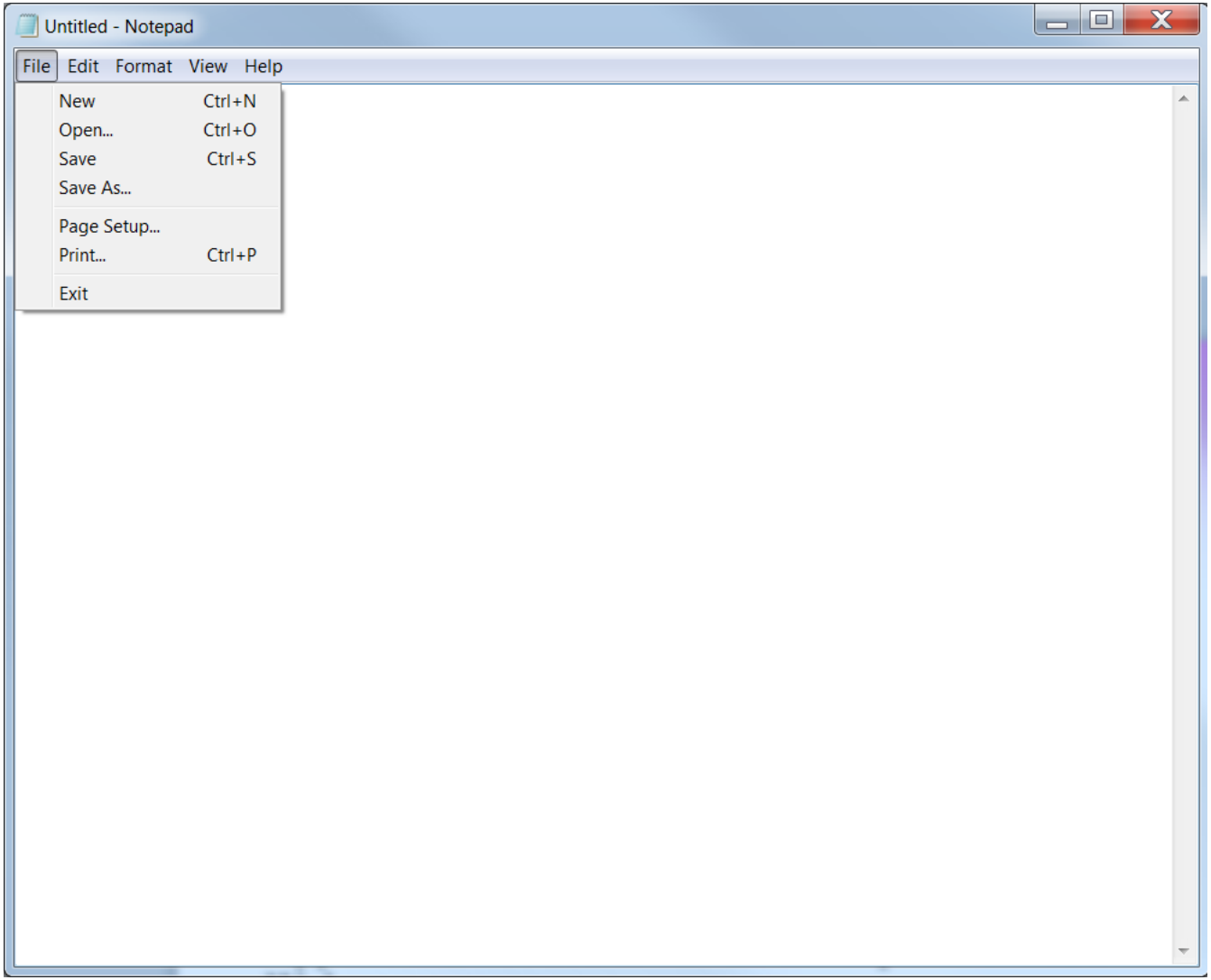
- The Keyboard
  - 1940
  - Key strokes

- The mouse
  - 1963
  - Move, click, hoover, scroll

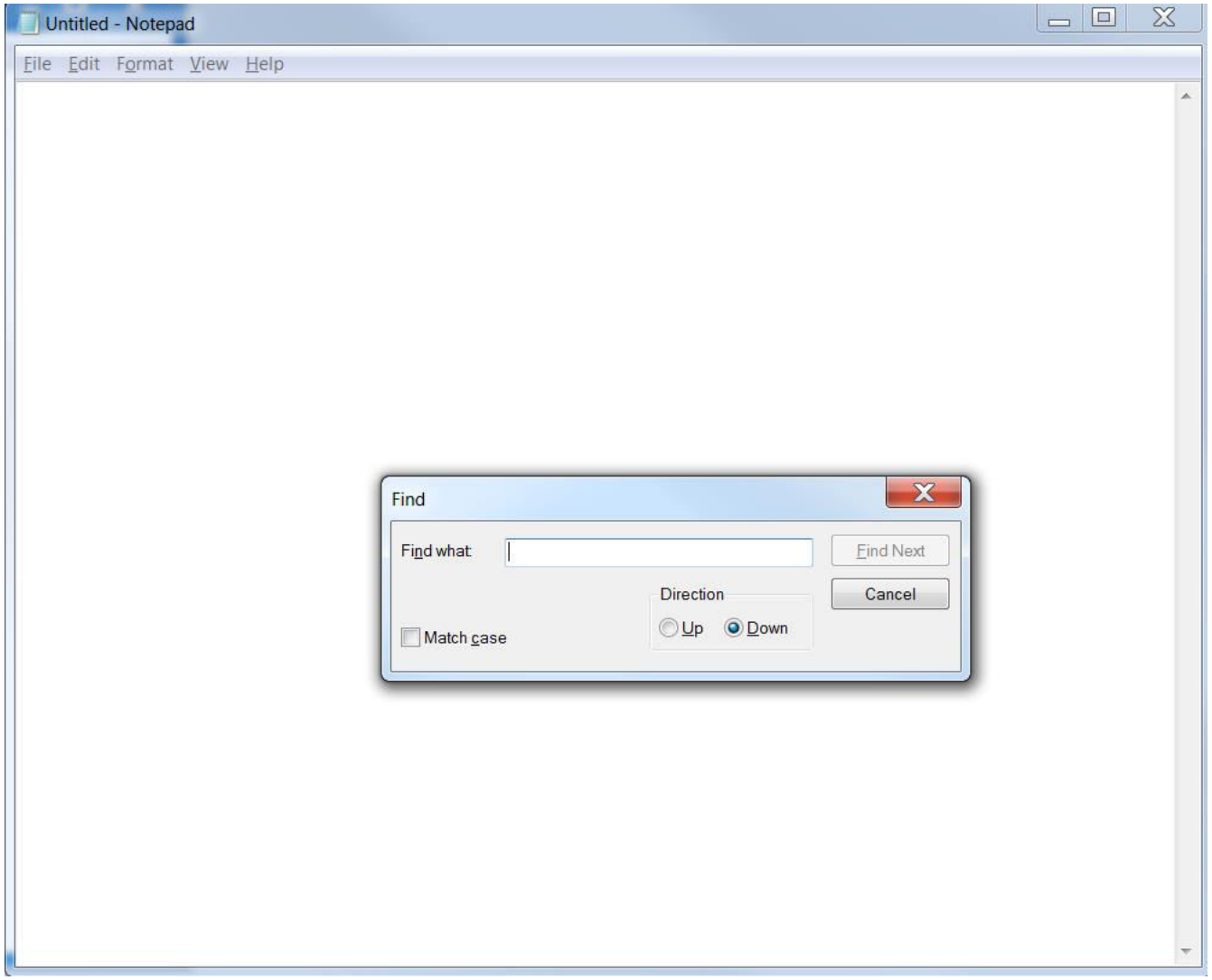
Window



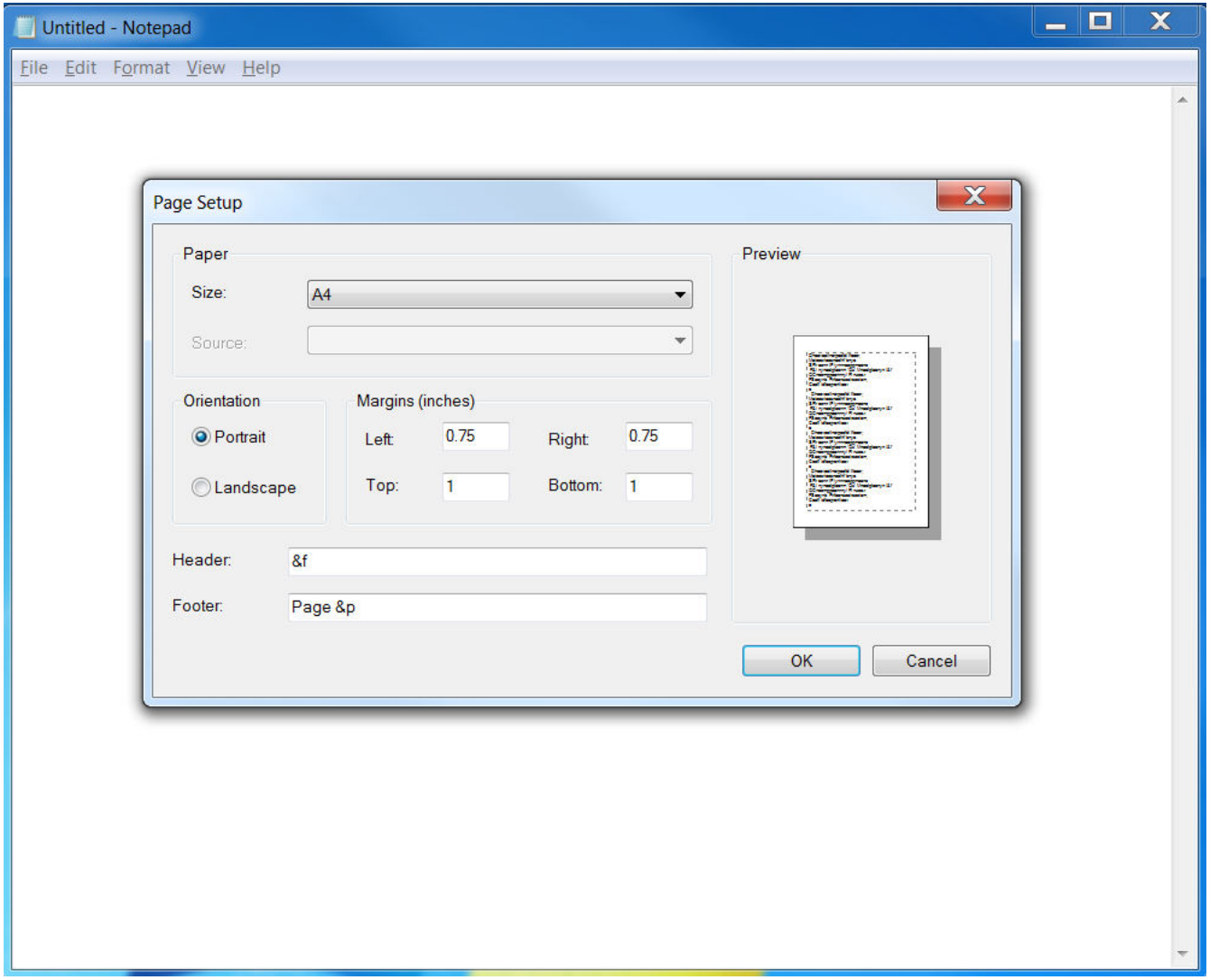
Menu


Untitled - Notepad
File
Edit
Format
View
Help
New Ctrl+N
Open... Ctrl+O
Save Ctrl+S
Save As...
Page Setup...
Print... Ctrl+P
Exit

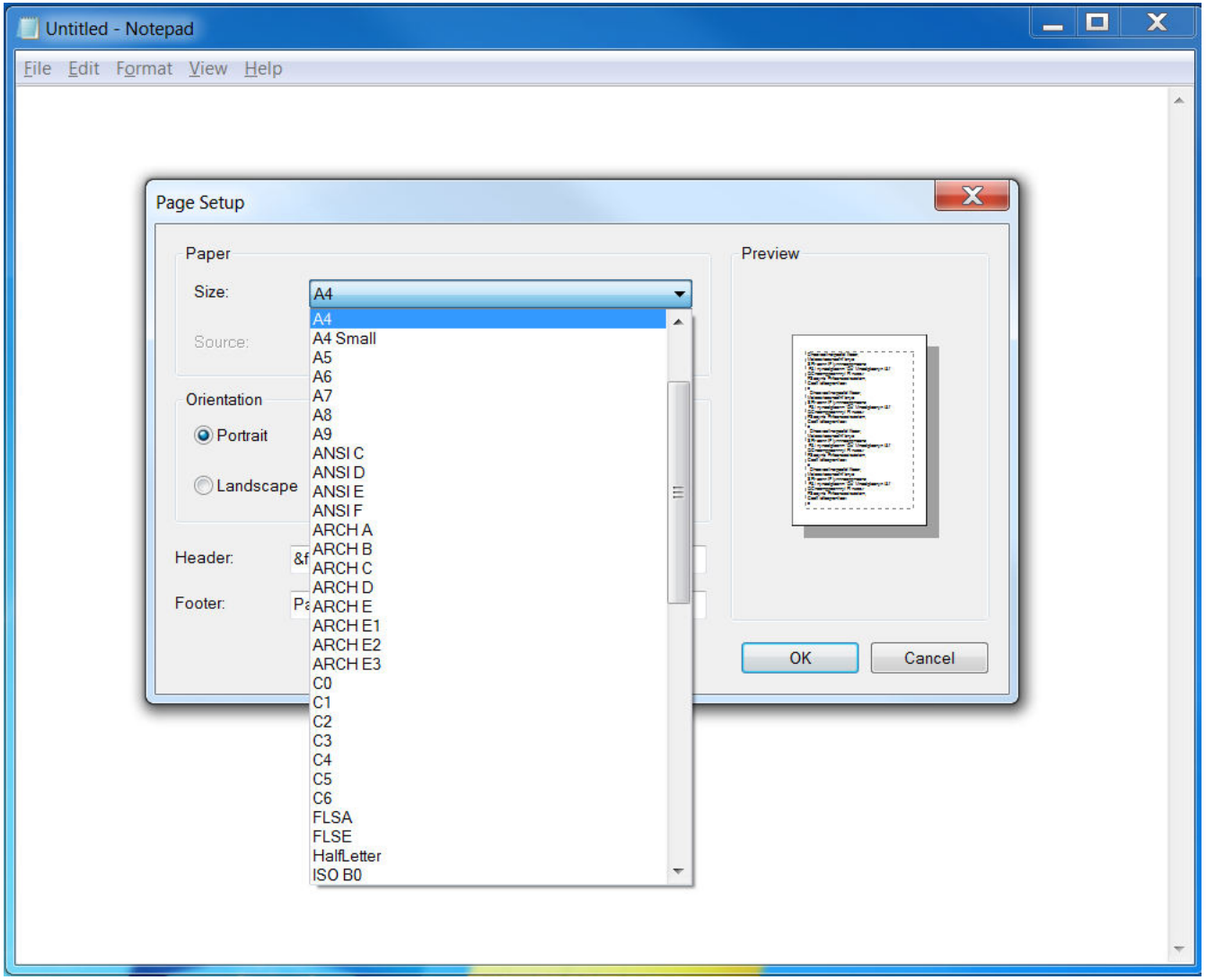
Dialog Box


Untitled - Notepad
File Edit Format View Help
Find
Find what:
Find Next
Cancel
Direction
Up
Down
Match case

Modal Dialog box

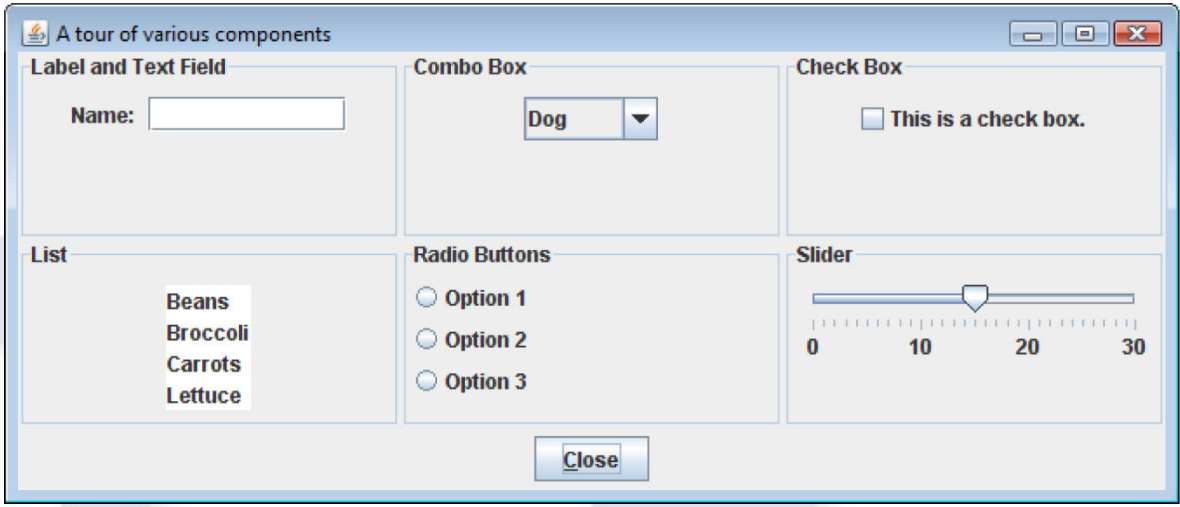


Combo Box


Untitled - Notepad
File Edit Format View Help
Page Setup
Paper
Size:
A4
Source:
Orientation
Portrait
Landscape
Header:
Footer:
Preview
OK
Cancel
A4
A4 Small
A5
A6
A7
A8
A9
ANSI C
ANSI D
ANSI E
ANSI F
ARCH A
ARCH B
ARCH C
ARCH D
ARCH E
ARCH E1
ARCH E2
ARCH E3
C0
C1
C2
C3
C4
C5
C6
FLSA
FLSE
HalfLetter
ISO B0

Components and Containers

- An application
  - A set of windows



- A window
  - A component itself
  - Contains a set of components

- A container
  - A component that contains other components
  - A window is a container

AWT vs. Swing

- AWT
  - Older
  - import java.awt.*
- **Swing**
  - Newer
  - import javax.swing.*

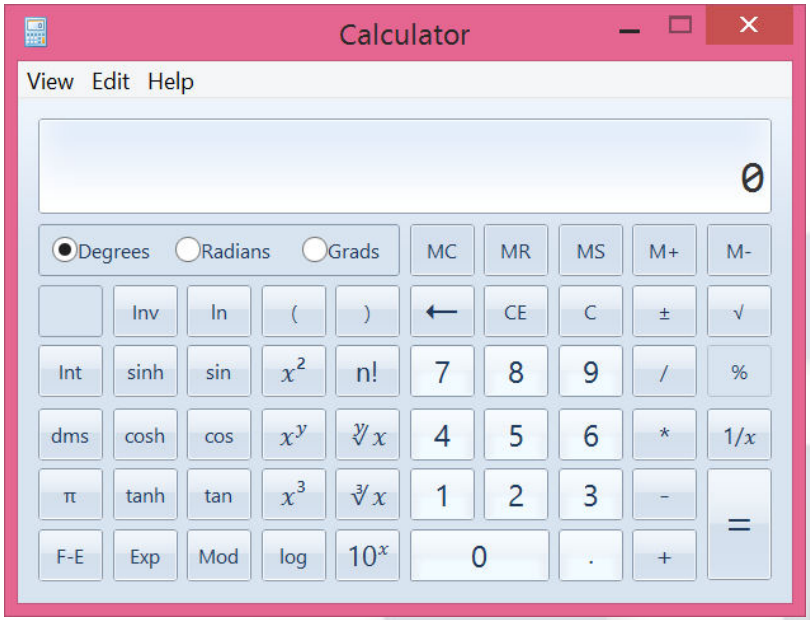
Creating a Window 1/2

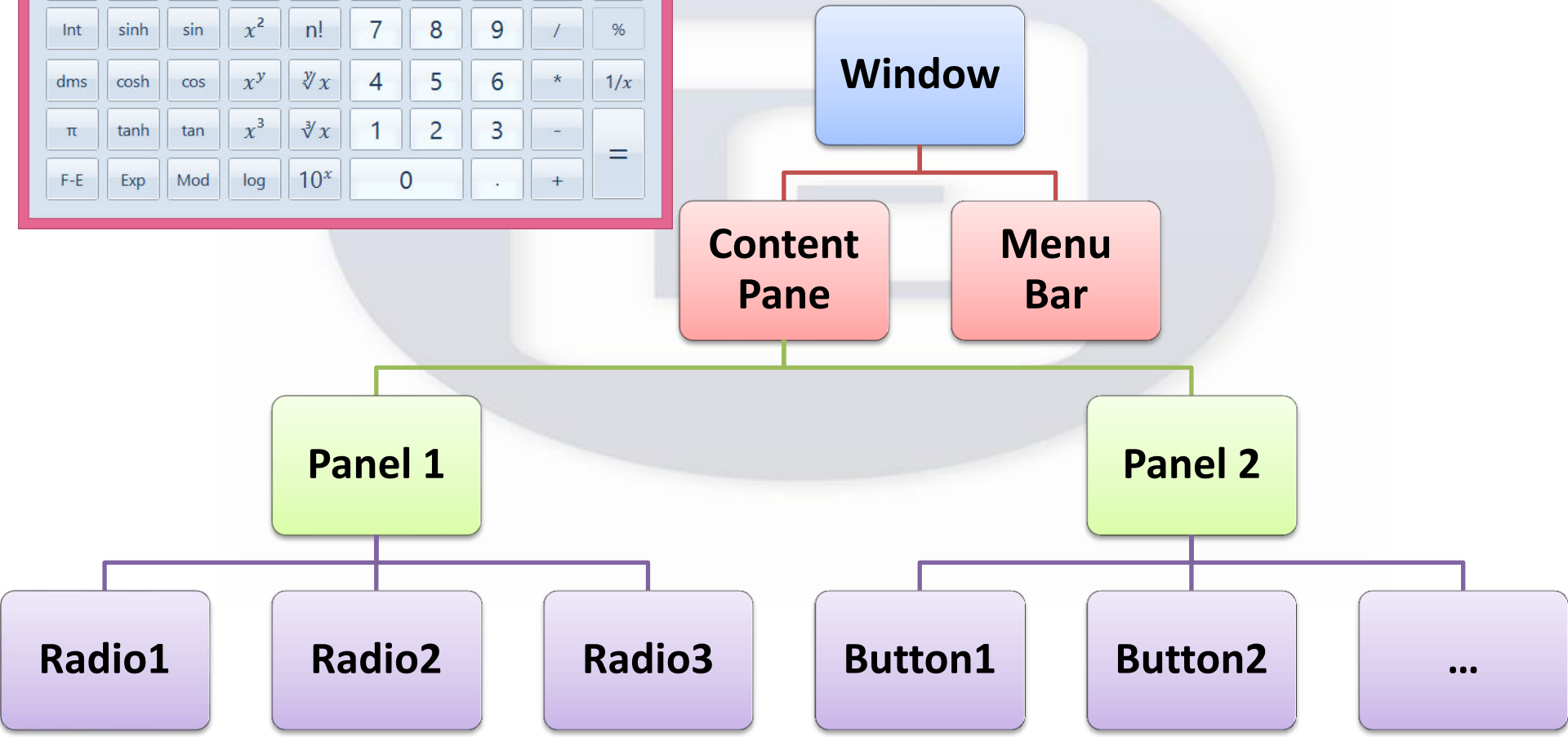
```java
import javax.swing.*;   // Needed for Swing classes

/**
   This class extends the JFrame class. Its constructor displays
   a simple window with a title. The application exits when the
   user clicks the close button.
*/

public class SimpleWindow extends JFrame
{
   /**
      Constructor
   */

   public SimpleWindow()
   {
      final int WINDOW_WIDTH = 350;   // Window width in pixels
      final int WINDOW_HEIGHT = 250;  // Window height in pixels

      // Set this window's title.
      setTitle("A Simple Window");

      // Set the size of this window.
      setSize(WINDOW_WIDTH, WINDOW_HEIGHT);

      // Specify what happens when the close button is clicked.
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      // Display the window.
      setVisible(true);
   }
```

Creating a Window 2/2

```
1  /**
2      This program creates an instance of the
3      SimpleWindow class.
4  */
5
6  public class SimpleWindowDemo
7  {
8      public static void main(String[] args)
9      {
10         SimpleWindow myWindow = new SimpleWindow();
11     }
12 }
```

Swing Hierarchy



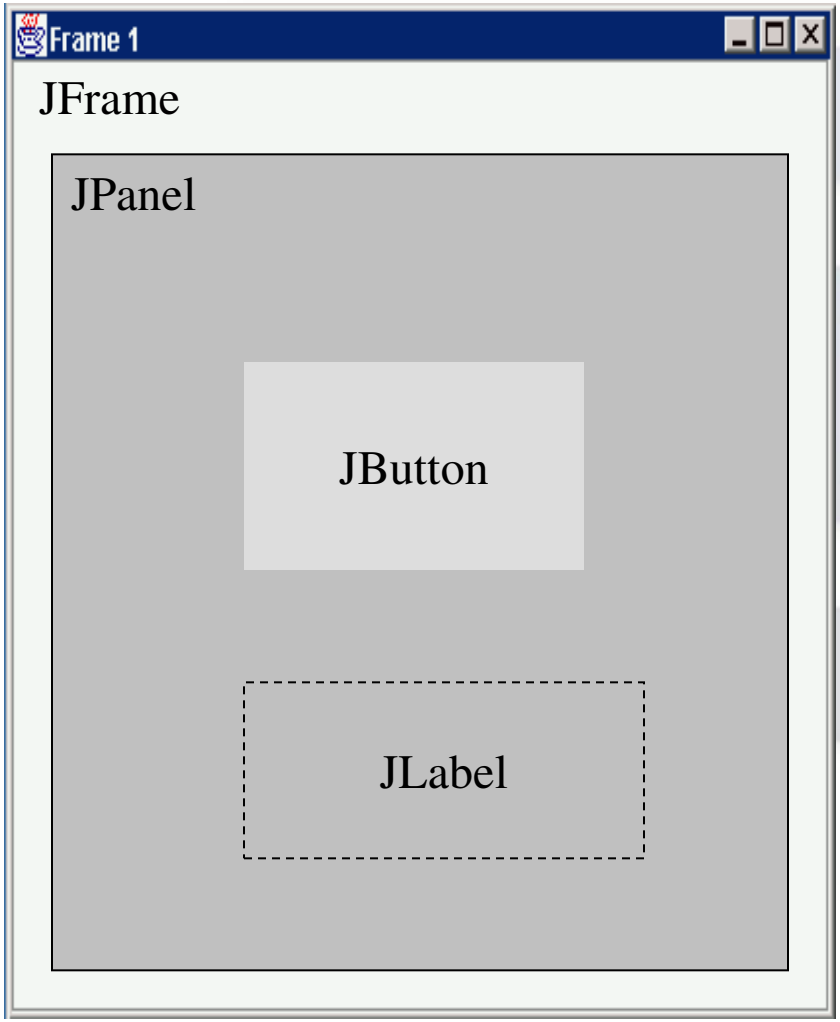


Panels and the content pane

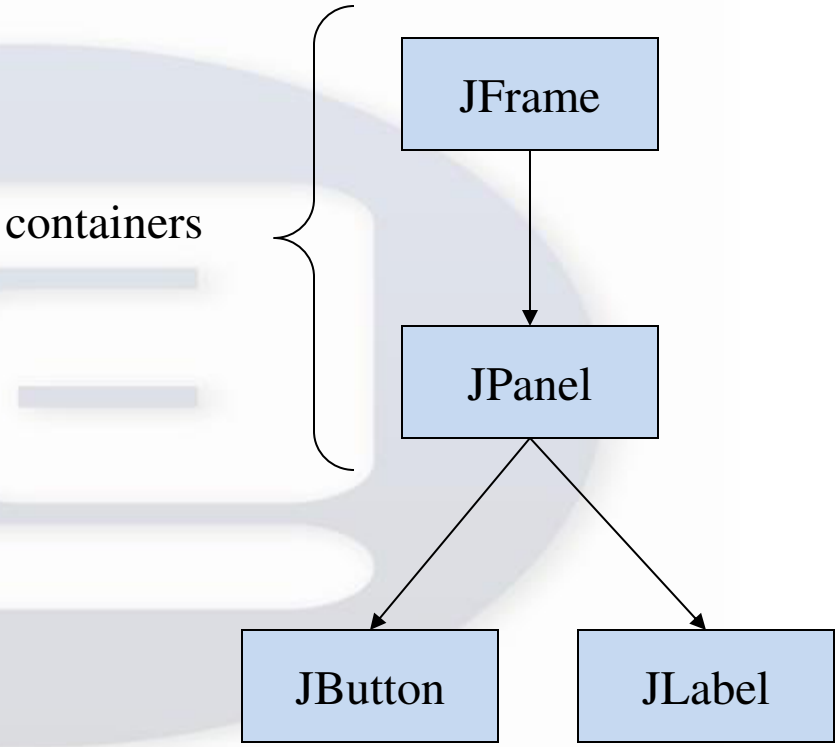
- Window
  - Visible
  - Has borders, title, and maximize/minimize buttons
- Content Pane
  - Invisible
  - Part of window
- Panel
  - Invisible container
  - Contains components
- Components
  - Visible
  - Textfield, button, radio…

Anatomy of an Application GUI

GUI

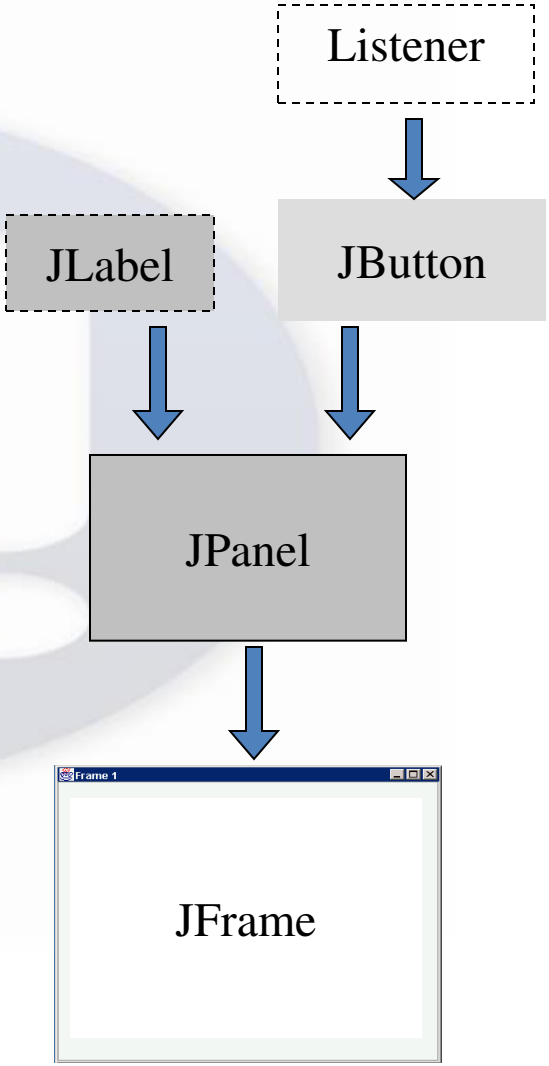


Internal structure



Build from bottom up

- Create:
  - Frame
  - Panel
  - Components
  - Listeners
- Add: (bottom up)
  - listeners into components
  - components into panel
  - panel into frame



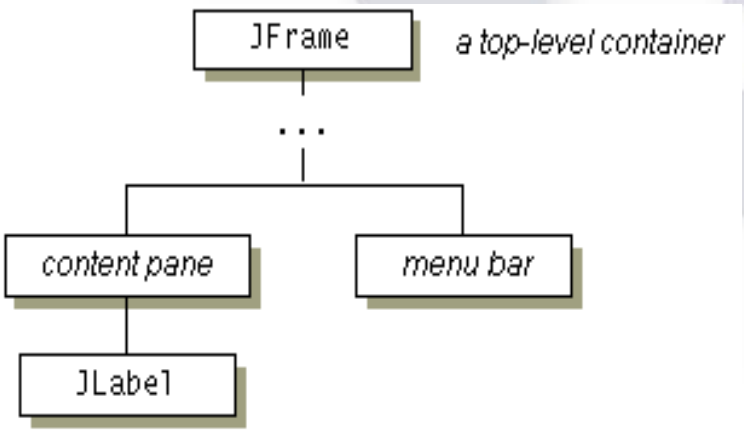
Containment hierarchy

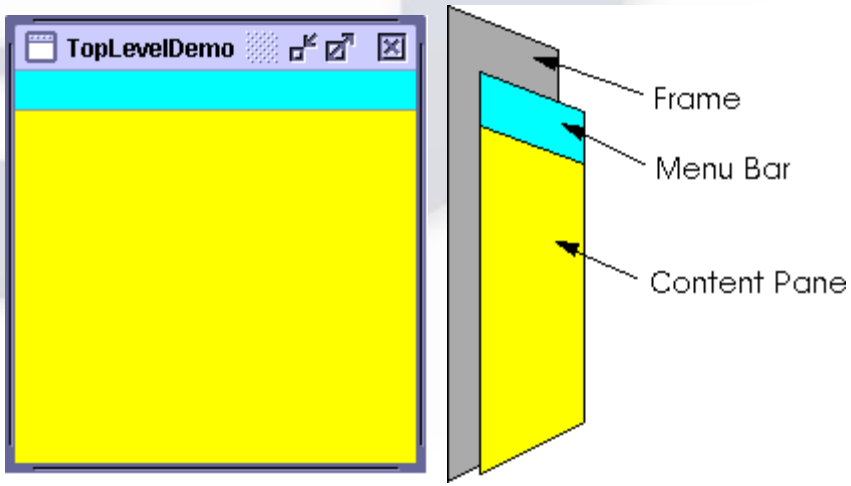
Top level containers: JFrame, JDialog, JApplet

Content pane: the main container in JApplet, JDialog, and JFrame Objects

Basic controls: JButton, JComboBox, List, Menu, Slider, JTextField, JLabel, progress bar, tool tip

General purpose containers: Panel, scroll pane, split pane, tabbed pane, tool bar

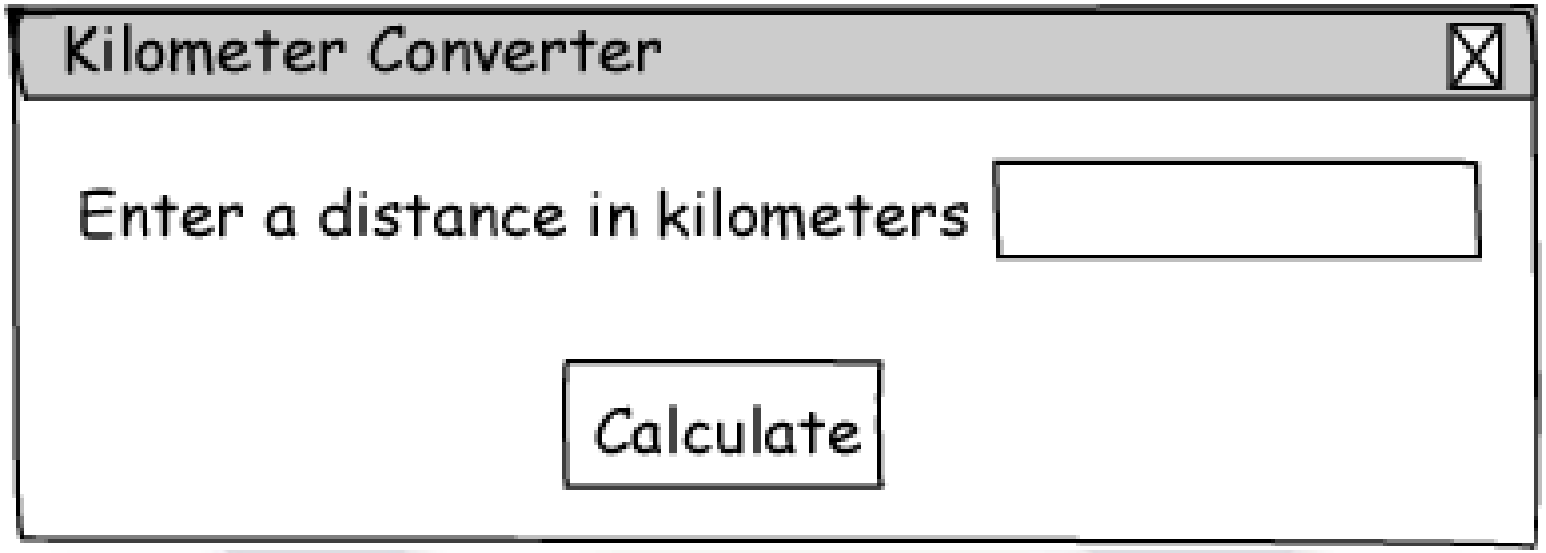




GUI Building Process

1. Create components (Button, Label, TextField)
2. Add components to a panel
3. Add the panel to the content pane of a window

Kilometer to Mile Converter

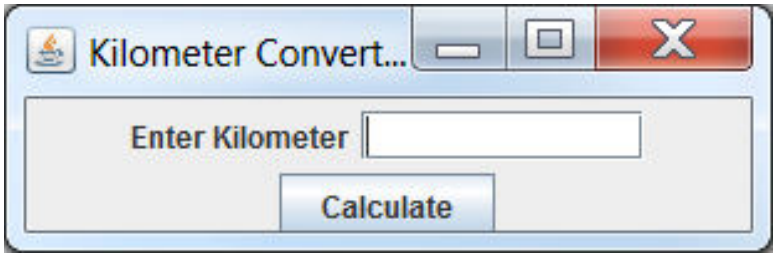


Kilometer to Mile Converter

```java
import javax.swing.*;

public class KiloConverterWindow extends JFrame
{
    private JPanel panel;
    private JLabel messageLabel;
    private JTextField kiloTextField;
    private JButton calcButton;
    private final int WINDOW_WIDTH = 310;
    private final int WINDOW_HEIGHT = 100;

public KiloConverterWindow()
    {
        // Window Setup
        setTitle("Kilometer Converter");
        setSize(WINDOW_WIDTH, WINDOW_HEIGHT);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Kilometer to Mile Converter

```
19        //Panel Setup
20        panel = new JPanel();
21        messageLabel = new JLabel("Enter Kilometer");
22        kiloTextField = new JTextField(10);
23        calcButton = new JButton("Calculate");
24
25        panel.add(messageLabel);
26        panel.add(kiloTextField);
27        panel.add(calcButton);
28
29        // Add the panel to the frame's content pane.
30        add(panel);
31
32        // Display the window.
33        setVisible(true);
34    }
35 }
```



Events

- Action that takes place
  - Mostly user action
- Mouse
  - Click/Right Click
  - Move/Hoover
- Keyboard
  - Key Press
- Other
  - Got Focus
  - Lost Focus

Events

- Event
  - User action
- Event Object
  - Contains information about the action
  - Generated by Java
- Event Source
  - The component that has the event
- Event Listener
  - Code that is executed if a certain event takes place

Listeners

1. import java.awt.event.*;
2. Class that implements ActionListener interface
3. Code actionPerformed method
4. Register with event source

Button Click Listener

2

3

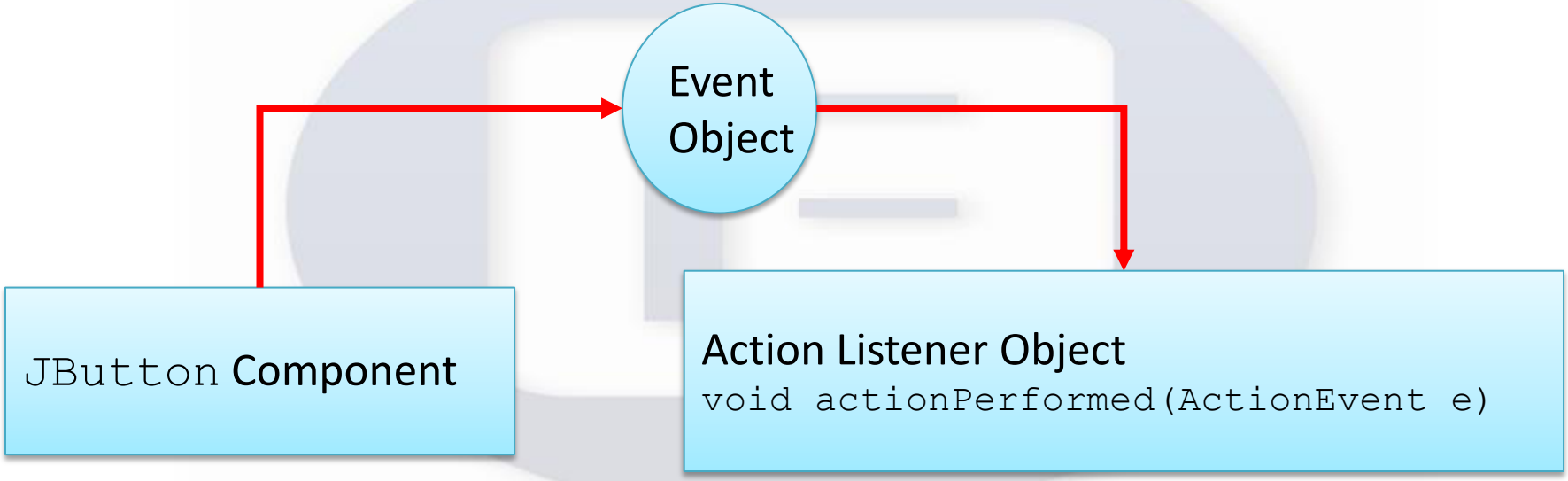
```
81    private class CalcButtonListener implements ActionListener
82    {
83
84       public void actionPerformed(ActionEvent e)
85       {
86          String input;  // To hold the user's input
87          double miles;  // The number of miles
88
89          // Get the text entered by the user into the
90          // text field.
91          input = kiloTextField.getText();
92
93          // Convert the input to miles.
94          miles = Double.parseDouble(input) * 0.6214;
95
96          // Display the result.
97          JOptionPane.showMessageDialog(null, input +
98                   " kilometers is " + miles + " miles.");
99       }
100   }
```

Kilometer to Mile Converter with Events

```
35      messageLabel = new JLabel("Enter a distance in kilometers");
36      kiloTextField = new JTextField(10);
37      calcButton = new JButton("Calculate");
38      calcButton.addActionListener(new CalcButtonListener());
39      panel = new JPanel();
40      panel.add(messageLabel);
41      panel.add(kiloTextField);
42      panel.add(calcButton);
```

4

Event Sequence



ActionEvent e

- Contains information about the event
- getActionCommand()
  - Event source, as a string
- getSource()
  - Event source, as an object

Layout Managers

- Why? [Problem]
  - Windows Resizing
  - Different Monitors with different resolutions
- What? [Solution]
  - Automatic resizing and repositioning of components
  - Layout managers
- Layout Managers:
  - GridLayout
  - BorderLayout
  - FlowLayout

Layout Managers

null

none,
programmer
sets x,y,w,h

FlowLayout

Left to right,
Top to bottom

GridLayout

BorderLayout

n
w
c
e
s

CardLayout

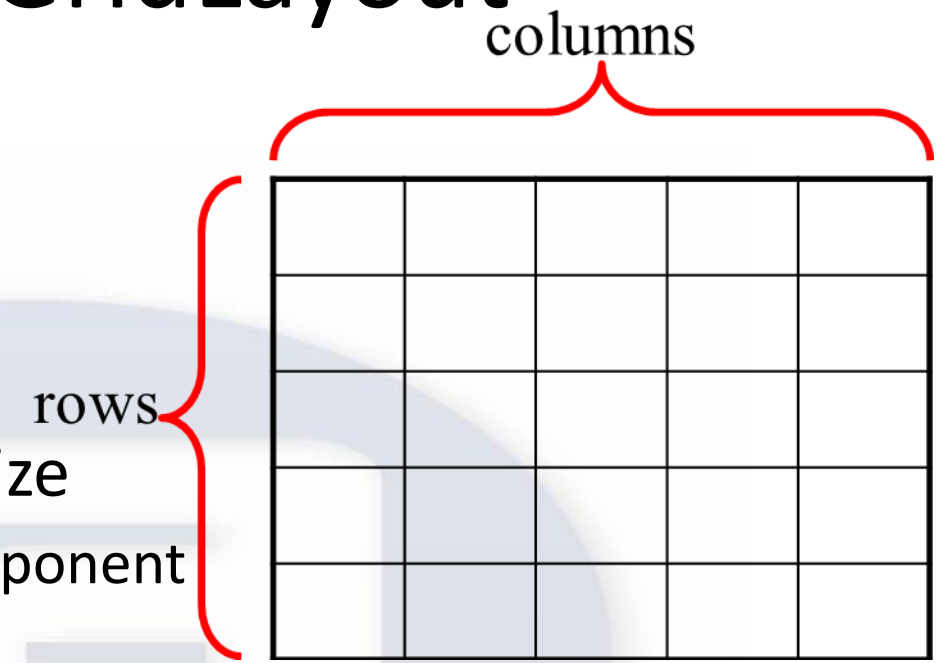
One at a time

GridBagLayout

JButton

GridLayout

- Grid of cells
  - Rows x Columns
  - One component per cell
  - All cells are of the same size
    - The size of the largest component



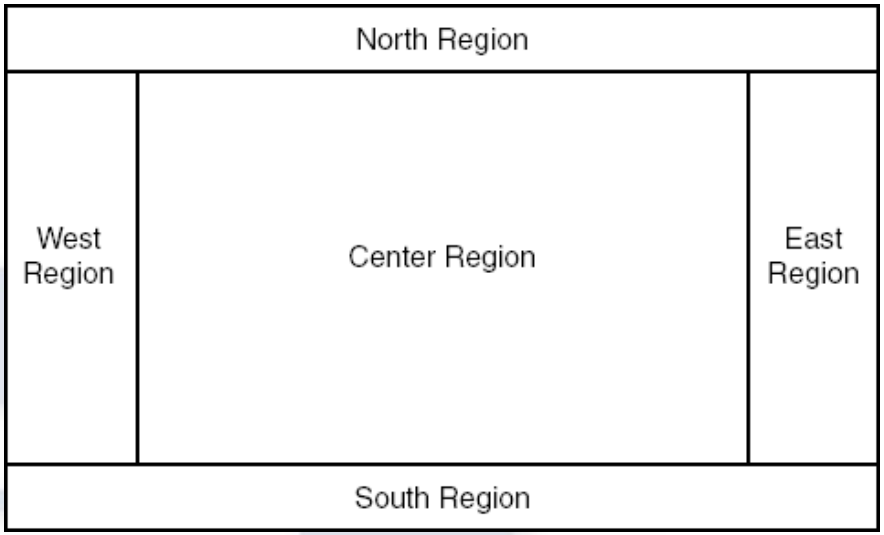
- GridLayout GL=new GridLayout (5, 5);
- setLayout (GL);
- add()
- add()
- add()

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15
16	17	18	19	20
21	22	23	24	25

GridLayout + Panels

- One component per cell is not enough!

BorderLayout



- Five regions
- One component per region
- Components are stretched

- JButton but = new JButton (“North!”);
- BorderLayout BL=new BorderLayout()
- add(but,BorderLayout.NORTH);
- setLayout(BL);

FlowLayout Manager

- Components added from left to right, top to bottom
- 5 pixels gaps between components
- Alignment
  - Left
  - Right
  - Center

Java GUI in a nutshell

1. Create an object of the required class
   - JLabel **Label**=new JLabel();
2. Create events handlers
   - private class **Listener** implements ActionListener {}
3. Set properties by invoking methods
   - **Label**.setText("This is a Label!");
4. Attach events handlers to the object
   - **Label**.addActionListener(**Listener**)
5. Manipulate the object via methods
   - String txt = **Label**.getText();
   - **Label**.setText("New Text");

Simple GUI-Based Input/Output with JOptionPane

- Dialog boxes
  - Used by applications to interact with the user
  - Provided by Java's `JOptionPane` class
    - Contains input dialogs and message dialogs

```
JOptionPane.showMessageDialog( parentWindow, String,
  title, messageType )
```

`parentWindow` - determines where dialog box appears

- **`null`** - displayed at center of screen
- window specified - dialog box centered horizontally over parent

```java
// Addition.java
// Addition program that uses JOptionPane for input and output.
import javax.swing.JOptionPane; // program uses JOptionPane

public class Addition
{
   public static void main( String args
   {
      // obtain user input from JOptionPane input dialogs
      String firstNumber =
         JOptionPane.showInputDialog( "Ente
      String secondNumber =
          JOptionPane.showInputDialog( "Enter second integer" );

      // convert String inputs to int values for use in a calculation
      int number1 = Integer.parseInt( firstNumber );
      int number2 = Integer.parseInt( secondNumber );

      int sum = number1 + number2; // add numbers

      // display result in a JOptionPane me
      JOptionPane.showMessageDialog( null, "The sum is " + sum,
         "Sum of Two Integers", JOptionPane.PLAIN_MESSAGE );
   } // end method main
} // end class Addition
```

Show input dialog to receive first integer

Show input dialog to receive second integer

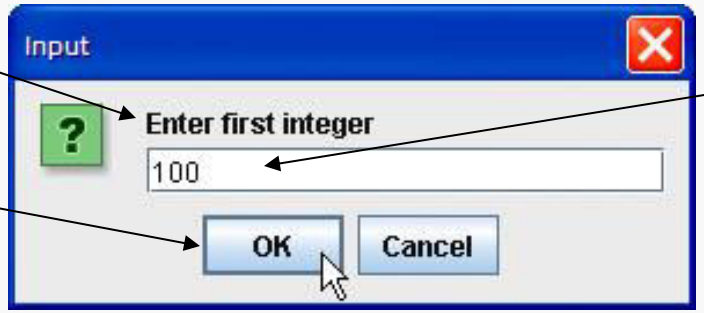
Show message dialog to output sum to user

Input dialog displayed by lines 10–11

Prompt to the user

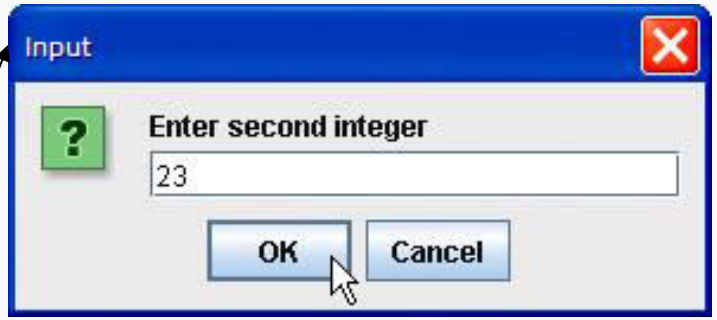
Text field in which the user types a value

When the user clicks **OK**, `showInputDialog` returns to the program the **100** typed by the user as a `String`. The program must convert the `String` to an `int`



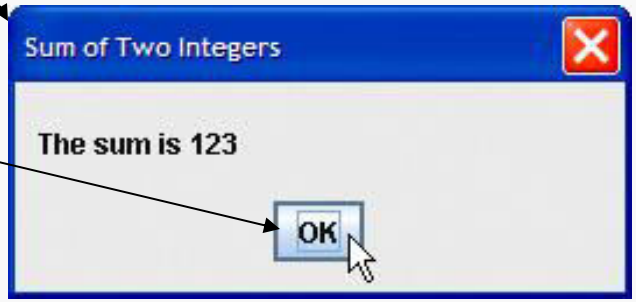
Input dialog displayed by lines 12—13

title bar



Message dialog displayed by lines 22—23

When the user clicks **OK**, the message dialog is dismissed (removed from the screen)



Message dialog type	Icon	Description
ERROR_MESSAGE		A dialog that indicates an error to the user.
INFORMATION_MESSAGE		A dialog with an informational message to the user.
WARNING_MESSAGE		A dialog warning the user of a potential problem.
QUESTION_MESSAGE		A dialog that poses a question to the user. This dialog normally requires a response, such as clicking a **Yes** or a **No** button.
PLAIN_MESSAGE	no icon	A dialog that contains a message, but no icon.

`JOptionPane` static constants for message dialogs.