the UNIVERSITY of GREENWICH

COMP1618
Lecture 5
Graphical User Interface (GUI)

the UNIVERSITY of GREENWICH

Lecture Objectives

- This lecture shows Java **GUI** (Graphic User Interface) objects:

  - Event-Driven Programming
  - Frame,
  - buttons
  - text fields,
  - combo boxes,
  - check boxes,
  - Layout manager

Event-Driven Programming Basics

- GUI programs usually use event-driven programming techniques.
- Basic idea behind event-driven programming:
  - The program waits for events to occur and then it responds.
- An *event* is a message that tells the program that something has happened.
  - For example, if the user clicks a button, then an event is generated, and it tells the program that a particular button was clicked.

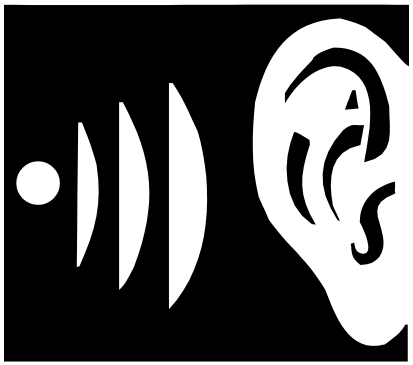
the UNIVERSITY of GREENWICH

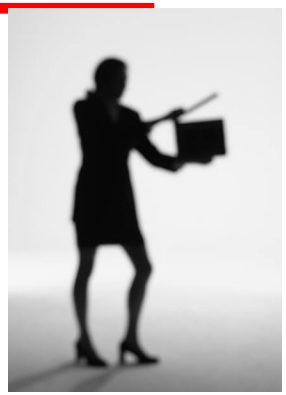
Event-Driven Programming Basics

- Note these additional event examples:

User Action	What Happens
Pressing the enter key while the cursor is inside a text box.	It tells the program that enter was pressed within the text box.
Clicking a menu item.	It tells the program that the menu item was selected.
Closing a window	It tells the program that the window's close button was clicked.

Event-Driven Programming Basics





An ***event*** occurs whenever an ***event listener*** detects an ***event trigger*** and responds by running a method called an ***event handler***.

Modern operating systems and programming languages contain facilities to let programmers set up event listeners.

the UNIVERSITY of GREENWICH

Designing Graphical User Interface (GUI)

AWT and Swing

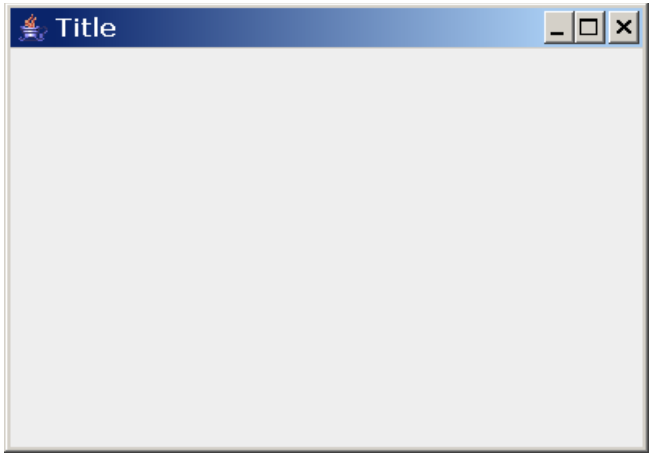
Java provides two sets of components for GUI programming:

AWT: classes in the **`java.awt`** package
Swing: classes in the **`javax.swing`** package

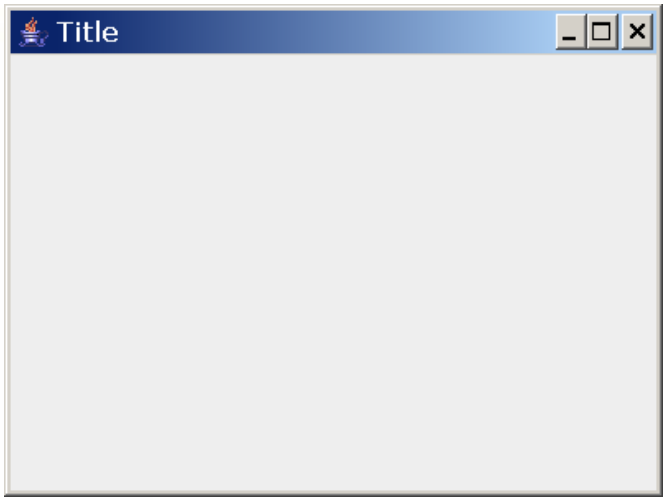
GUI components: JFrame

- **JFrame** is a top-level container that provides a window on the screen.

- It belongs to Java Swing library
  - `import javax.swing.*;`



```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class MyJF extends JFrame
6                 implements ActionListener
7  {
8
9     public static void main(String[] args)
10    {
11       MyJF jf = new MyJF();
12    }
13
14    public MyJF()
15    {
16       setLayout(new FlowLayout());
17       setSize(400, 300);
18       setTitle("Title");
19       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20       setVisible(true);
21    }
22
23    public void actionPerformed(ActionEvent e)
24    {
25       // add your event handling code here
26    }
27 }
```



constructor

Buttons

`JButton` is a class in package `javax.swing` that represents buttons on the screen.

The most common constructor is:

```
public JButton(String label);
```

E.g.:

```
JButton myButton = new JButton("Text");
```

Adding a button- 3 parts

1

```
JButton myButton = new JButton("Press");
```

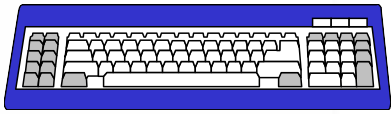
2

```
add(myButton);

myButton.addActionListener(this);
```

3

```
public void actionPerformed(ActionEvent e)
    {
      setTitle("pressed");
    }
```

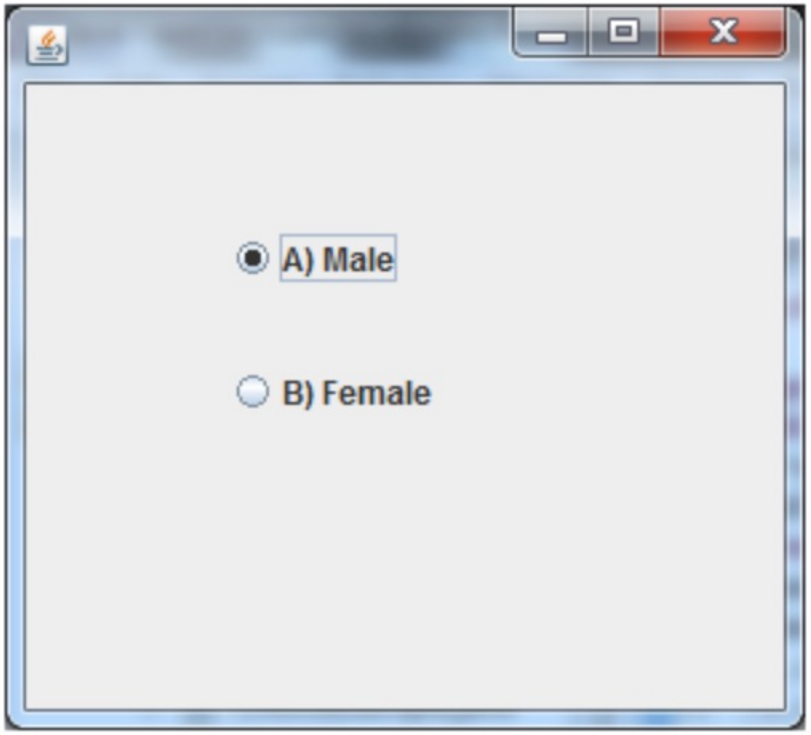
```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class FirstFrame2 extends JFrame
6              implements ActionListener
7  {
8
9    JButton myButton = new JButton("Press");
10
11     public static void main(String[] args)
12     {
13        FirstFrame2 ff = new FirstFrame2();
14     }
15
16     public FirstFrame2()
17     {
18        setLayout(new FlowLayout());
19        setSize(400, 400);
20        setTitle("Button");
21        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22        add(myButton);
23        myButton.addActionListener(this);
24
25        setVisible(true);
26     }
27
28     public void actionPerformed(ActionEvent e)
29     {
30       setTitle("pressed");
31     }
32 }
```



FirstFrame2.java

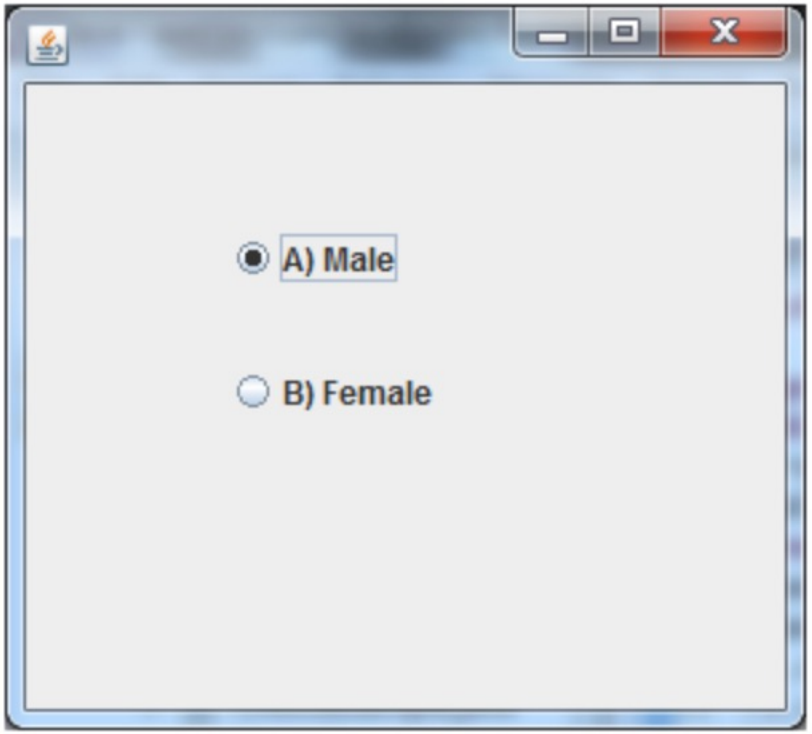
Radio Button

The `JRadioButton` class is used to create a radio button. It is used to choose one option from multiple options.



Example: Radio Button

```
import javax.swing.*;
public class RadioButtonExample {
JFrame f;
RadioButtonExample(){
f=new JFrame();
JRadioButton r1=new JRadioButton("A) Male");
JRadioButton r2=new JRadioButton("B) Female");
r1.setBounds(75,50,100,30);
r2.setBounds(75,100,100,30);
ButtonGroup bg=new ButtonGroup();
bg.add(r1);bg.add(r2);
f.add(r1);f.add(r2);
f.setSize(300,300);
f.setLayout(null);
f.setVisible(true);
}
public static void main(String[] args) {
   new RadioButtonExample();
}
}
```



Text Field

A `JTextField` is an area that the user can type one line of text into. It is a good way of getting text input from the user.

```
JTextField myText = new JTextField(10);
```

Adding a TextField- 2 parts

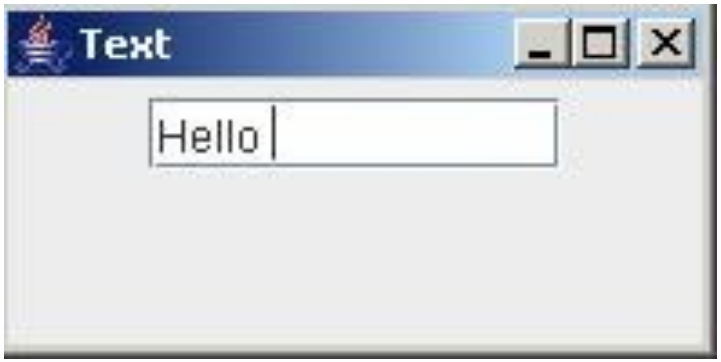
1

```
JTextField myText = new JTextField(10);
```

2

```
add(myText);
```

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

public class HelloText extends JFrame
            implements ActionListener
{
   JTextField myText = new JTextField(10);


   public static void main(String[] args)
   {
      HelloText jt = new HelloText();
   }

   public HelloText()
   {
      setLayout(new FlowLayout());
      setSize(200, 100);
      setTitle("Text");
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      add(myText);


      setVisible(true);
   }

   public void actionPerformed(ActionEvent e)
   {

   }
```



the UNIVERSITY of GREENWICH

Text Area

- A text area is less restrictive than a text field and allows us to have quite a bit of text. It is instantiated using

  JTextArea myArea = new JTextArea(int, int);

- The two integers in this constructor control the number of lines and the width in characters of the text area

```java
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class TextAreaDemo extends JFrame
6              implements ActionListener
7  {
8     JTextField nameTxt = new JTextField(10);
9     JTextArea output = new JTextArea(2, 30);
10    JButton sub = new JButton("Submit");
11
12    public static void main(String[] args)
13    {
14       TextAreaDemo jf = new TextAreaDemo();
15    }
16
17    public TextAreaDemo()
18    {
19       setLayout(new FlowLayout());
20       setSize(400, 120);
21       setTitle("Text Area Demo");
22       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23
24       add(new Label("Type your name:"));
25       add(nameTxt);
26       add(sub);
27       sub.addActionListener(this);
28       add(output);
29       output.setEditable(false);
30       setVisible(true);
31    }
32
33    public void actionPerformed(ActionEvent e)
34    {
35       String name = nameTxt.getText();
36       String message = "Hello " + name + " \nEnjoy your programming ";
37       output.setText(message);
38    }
39 }
```

creating 3 objects
a text field, a text area
and a button

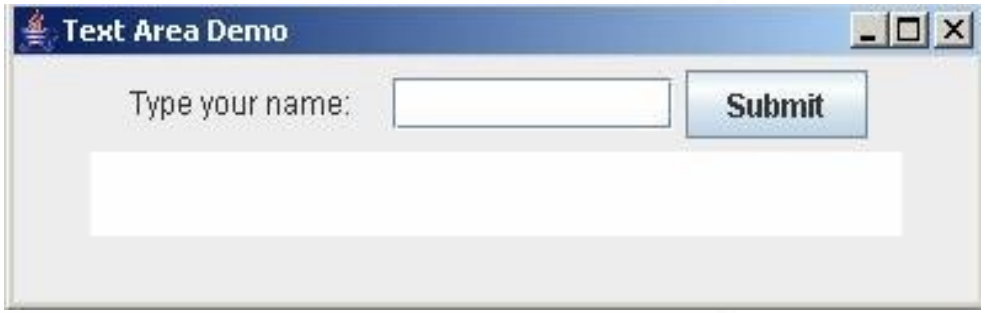
adding the objects to the content pane and registering the button with the ActionListener

using the getText methods form the JTextField class to give the String in the text field
to the variable name

creating the message to send to the text area output

using the setText method of JTextArea class

```java
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class TextAreaDemo extends JFrame
6              implements ActionListener
7  {
8     JTextField nameTxt = new JTextField(10);
9     JTextArea output = new JTextArea(2, 30);
10    JButton sub = new JButton("Submit");
11
12    public static void main(String[] args)
13    {
14       TextAreaDemo jf = new TextAreaDemo();
15    }
16
17    public TextAreaDemo()
18    {
19       setLayout(new FlowLayout());
20       setSize(400, 120);
21       setTitle("Text Area Demo");
22       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23
24       add(new Label("Type your name:"));
25       add(nameTxt);
26       add(sub);
27       sub.addActionListener(this);
28       add(output);
29       output.setEditable(false);
30       setVisible(true);
31    }
32
33    public void actionPerformed(ActionEvent e)
34    {
35       String name = nameTxt.getText();
36       String message = "Hello " + name + " \nEnjoy your programming ";
37       output.setText(message);
38    }
```



CheckBox

- The `JCheckBox` class is used to create a checkbox. It is used to turn an option on (true) or off (false).

- They can easily be used for
  - *yes/no*
  - *male/female*
- The code for making a new check box object is

```
JCheckBox myCheckBox = new JCheckBox("Text");
```

Adding a CheckBox- 3 parts

```
JCheckBox myChBx = new JCheckBox("Try checking it");
```

create a checkbox object called myChBx with the text **Try checking it** at its side

```
add(myChBx);

myChBx.addActionListener(this);
```

set the action to be triggered if someone checks it

```
public void actionPerformed(ActionEvent e)
    {
        myTxt.setText("you did it!");
    }
```

message to appear in a text field if box is ticked

```java
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class CheckText extends JFrame
6                 implements ActionListener
7  {
8     JTextField myTxt = new JTextField(10);
9     JCheckBox myChBx = new JCheckBox("Try checking it");
10
11    public static void main(String[] args)
12    {
13       CheckText jf = new CheckText();
14    }
15
16    public CheckText()
17    {
18       setLayout(new FlowLayout());
19       setSize(600, 120);
20       setTitle("Check Box Example");
21       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22
23       add(myTxt);
24       add(myChBx);
25       myChBx.addActionListener(this);
26       setVisible(true);
27    }
28
29    public void actionPerformed(ActionEvent e)
30    {
31      myTxt.setText("you did it!");
32     }
33 }
```

Slightly more complex

If we extend this example to give us **2 check boxes** we come across several problems:

- How do we know **which one** is checked?

How do we know which one is chosen?

```
public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() == myRedChBx)
        {
            myTxt.setText("you have chosen Red");
        }
        else if (e.getSource() == myGreenChBx)
        {
            myTxt.setText("you have chosen Green");
        }
 }
```

actionPerformed method

the UNIVERSITY of GREENWICH

How do we know which one is chosen?

Alternatively we can use the test if (myChBx.isSelected())

```
public void actionPerformed(ActionEvent e)
{
  if (myRedChBx.isSelected())
  {
        myTxt.setText("you have chosen Red");
  }
  else if (myGreenChBx.isSelected())
  {
        myTxt.setText("you have chosen Green");
  }
}
```

Labels

A `JLabel` is a component which contains a string. The constructors are

```
public JLabel() // creates label with no text
public JLabel(String text) //create label with text
```

The methods available are

```
public String getText() // return label text
public void setText(String s) // sets the label text
```

Add to an existing frame

```
add(new Label("String"));
```

```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class LabelDemo extends JFrame
6              implements ActionListener
7  {
8     JTextField nameTxt = new JTextField(10);
9     JButton sub = new JButton("Submit");
10
11    public static void main(String[] args)
12    {
13       LabelDemo jf = new LabelDemo();
14    }
15
16    public LabelDemo()
17    {
18       setLayout(new FlowLayout());
19       setSize(400, 100);
20       setTitle("Label Demo");
21       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
22
23       add(new Label("Type your name:"));
24       add(nameTxt);
25       add(sub);
26       sub.addActionListener(this);
27       setVisible(true);
28       setResizable(false);
29    }
30
31    public void actionPerformed(ActionEvent e)
32    {
33       nameTxt.setText("");
34    }
```

label added here

only listening for the button using keyboard return in the text field has no effect

pressing submit clears the text field

Combo Boxes

- `JComboBox` provides a pop-up list of items from which the user can make a selection.

- The constructor is:

```
public JComboBox() // create new choice button
```

- The most useful methods are:

```
public void addItem(Object s) // add s to list of choices
public int getItemCount() // return # choices in list
public Object getItemAt(int index) // return item at index
public Object getSelectedItem() // return selected item
public int getSelectedIndex() // return index of selected
```

How do we put items in the box?

Add Strings to the drop down box using the Java method addItem

- if we have the definition

```
JComboBox myCBox = new JComboBox();
```

- then we have statements like

```
myCBox.addItem("Item1");
myCBox.addItem("Item2");
myCBox.addItem("Item3");
```

We use the method getSelected to find out which one has been chosen

```
if (myCBox.getSelectedItem()=="Item2")
```

Adding a Combo Box 3 parts

```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class SimpleCombo extends JFrame
6              implements ActionListener
7  {
8    JComboBox colour = new JComboBox();
9    JLabel instrLabel = new JLabel("Choose a colour from the drop down menu:  ");
10   JTextField colourTxt = new JTextField(15);
11    JButton doneBtn = new JButton("Finish");
12
13    public static void main(String[] args)
14    {
15       SimpleCombo sc = new SimpleCombo();
16    }
17
18    public SimpleCombo()
19    {
20       setLayout(new FlowLayout());
21       setSize(400, 150);
22       setTitle("Colours");
23       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
24       colour.addItem("red");
25       colour.addItem("yellow");
26       colour.addItem("blue");
27       colour.addItem("green");
28       add(instrLabel);
29       add(colour);
30       add(colourTxt);
31       add(doneBtn);
32       doneBtn.addActionListener(this);
33       setVisible(true);
34    }
35
36    public void actionPerformed(ActionEvent e)
37    {
38       if (colour.getSelectedItem()=="red") colourTxt.setText("for poppies");
39       else if (colour.getSelectedItem()=="yellow") colourTxt.setText (" like the sun");
40       else if (colour.getSelectedItem()=="blue") colourTxt.setText( " like the sky ");
41       else  colourTxt.setText(" like the grass");
42    }
43 }
```

make a ComboBox

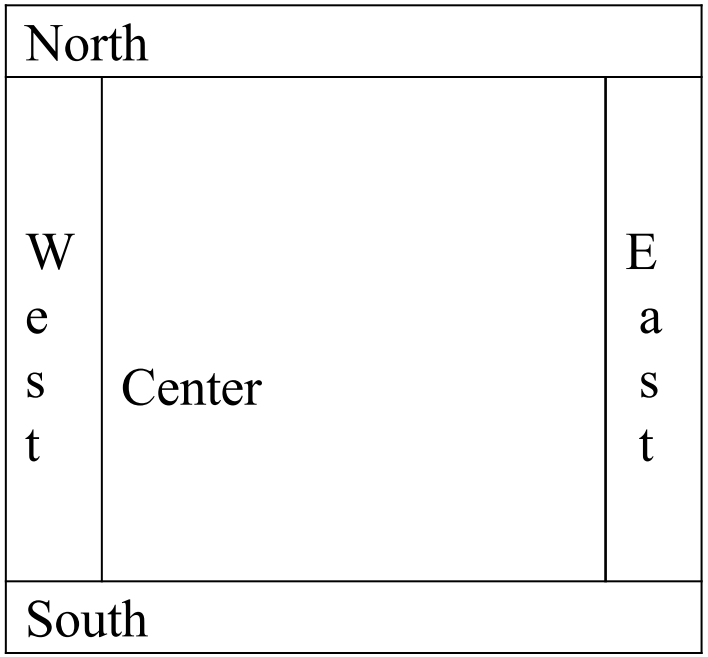
add the items to the ComboBox and the ComboBox to the content pane

have actions triggered by button and dependent on the choice made

different text displayed
in the text field according
to the selection made

Layout Managers

The **LayoutManagers** are used to arrange components in a particular manner. The Java LayoutManagers facilitates us to control the positioning and size of the components in GUI forms.



java.awt.BorderLayout

- Border layout splits the content pane (or other container such as a *panel*) into up to five regions North, South, East, West and Center

ComboText.java

```
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class ComboText extends JFrame
6                 implements ActionListener
7  {
8     JComboBox tut = new JComboBox();
9     JTextArea commentTxt = new JTextArea(2,15);
10    JButton doneBtn = new JButton("Finish");
11
12    public static void main(String[] args) { new ComboText(); }
13
14    public ComboText()
15    {
16       setLayout(new BorderLayout());
17       setSize(400, 150);
18       setTitle("ComboText Demo");
19       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20      JPanel top = new JPanel();
21       top.setLayout(new FlowLayout());
22       tut.addItem("Don");
23       tut.addItem("Kate");
24       tut.addItem("Chris");
25       top.add (new Label("choose your favourite tutor"));
26       top.add (tut);
27       add("North", top);
28       add("Center", commentTxt);
29       add("South", doneBtn);
30       doneBtn.addActionListener(this);
31       setVisible(true);
32    }
33
34    public void actionPerformed(ActionEvent e)
35    {
36       String com;
37       if (tut.getSelectedItem()=="Don") com = " you must like programming";
38       else if (tut.getSelectedItem()=="Kate") com = " you think that will get you e
39       else  com = " you did not say which Chris";
40       commentTxt.setText(" I see that "+ com);
41    }
42 }
```

making 3 objects
a combo box , a text area and a button

note short version of method

giving the frame borders

making a panel with FlowLayout for the North border to organise the label & combo box

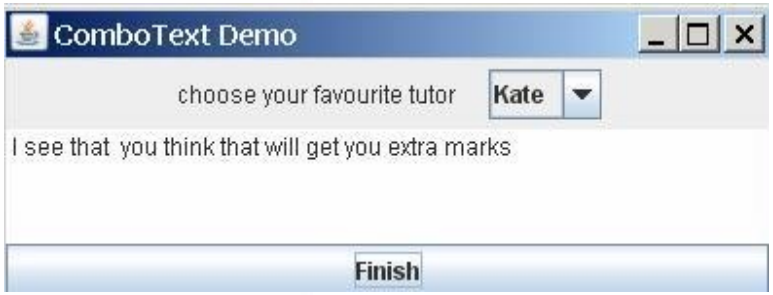
adding the items to the combo box

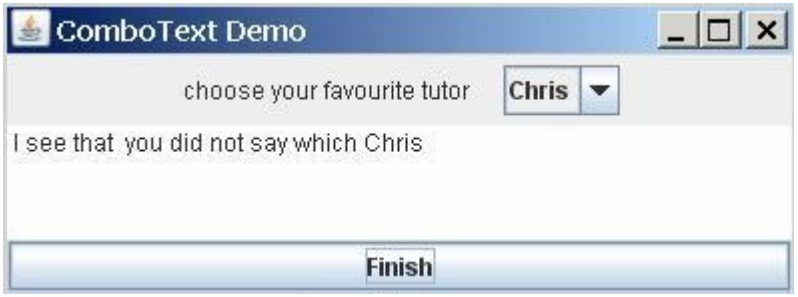
putting the label & the combo box on the panel then adding the panel to the North border

putting the text area in the Center border and the button on the South border

in response to a click on the Finish button a different comment is put depending on the combo box selection

```
1  import java.awt.*;
2   import javax.swing.*;
3   import java.awt.event.*;
4
5   public class ComboText extends JFrame
6                 implements ActionListener
7   {
8      JComboBox tut = new JComboBox();
9      JTextArea commentTxt = new JTextArea(2,15);
10     JButton doneBtn = new JButton("Finish");
11
12     public static void main(String[] args) { new ComboText(); }
13
14     public ComboText()
15     {
16        setLayout(new BorderLayout());
17        setSize(400, 150);
18        setTitle("ComboText Demo");
19        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
20       JPanel top = new JPanel();
21        top.setLayout(new FlowLayout());
22        tut.addItem("Don");
23        tut.addItem("Kate");
24        tut.addItem("Chris");
25        top.add (new Label("choose your favourite tutor"));
26        top.add (tut);
27        add("North", top);
28        add("Center", commentTxt);
29        add("South", doneBtn);
30        doneBtn.addActionListener(this);
31        setVisible(true);
32     }
33
34     public void actionPerformed(ActionEvent e)
35     {
36        String com;
37        if (tut.getSelectedItem()=="Don") com = " you must like programming";
38        else if (tut.getSelectedItem()=="Kate") com = " you think that will get you extra marks";
39        else  com = " you did not say which Chris";
40        commentTxt.setText(" I see that "+ com);
41     }
42  }
```



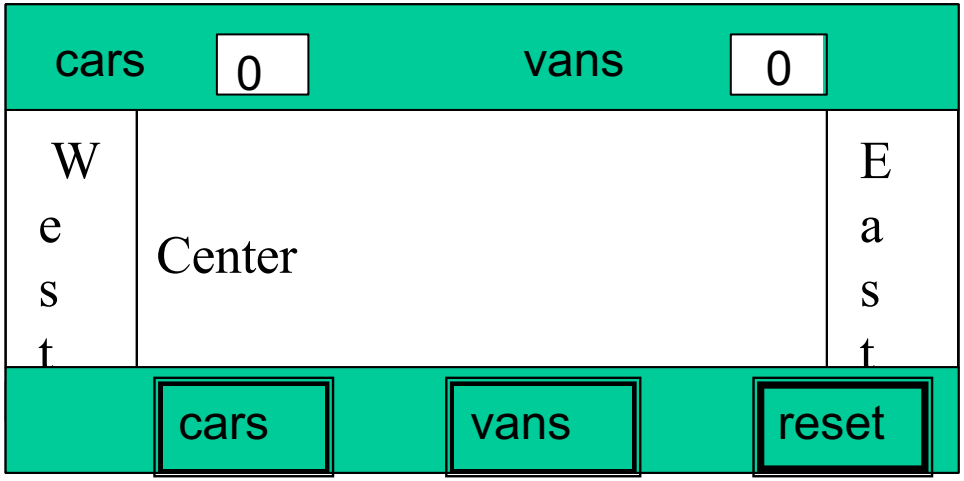


Java FlowLayout

The **FlowLayout** class is used to arrange the components in a line, one after another (in a flow)

An example: FlowLayout

The **FlowLayout** class is used to arrange the components in a line, one after another (in a flow)




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

bottom.add(carsBtn);
bottom.add(vansBtn);
bottom.add(reset);
carsBtn.addActionListener(this);
vansBtn.addActionListener(this);
reset.addActionListener(this);
```

```java
1  import java.awt.*;
2  import javax.swing.*;
3  import java.awt.event.*;
4
5  public class Vehicles extends JFrame
6              implements ActionListener
7  {
8     JTextField carsTxt = new JTextField(3);
9     JTextField vansTxt = new JTextField(3);
10    JButton carsBtn = new JButton("Car");
11    JButton vansBtn = new JButton("Van");
12    JButton reset = new JButton("Reset");
13    int cars = 0, vans = 0; // counters
14
15    public static void main(String[] args)
16    {
17    new Vehicles();
18    }
19
20    public Vehicles()
21    {
22       setLayout(new BorderLayout());
23       setSize(400, 140);
24       setTitle("Car and van counter");
25       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
26       JPanel top = new JPanel();
27       top.setLayout(new FlowLayout());
28       top.add(new Label("Cars:"));
29       top.add(carsTxt); carsTxt.setEditable(false);
30       top.add(new Label("        Vans:"));
31       top.add(vansTxt); vansTxt.setEditable(false);
32       carsTxt.setText("0");
33       vansTxt.setText("0");
34       add("North", top);
35       JPanel bottom = new JPanel();
36       bottom.setLayout(new FlowLayout());
37       bottom.add(carsBtn);
38       bottom.add(vansBtn);
39       bottom.add(reset);
40       carsBtn.addActionListener(this);
41       vansBtn.addActionListener(this);
42       reset.addActionListener(this);
43       add("South", bottom);
44       setVisible(true);
45    }
46
47    public void actionPerformed(ActionEvent e)
48    {
49       if (e.getSource() == carsBtn)     cars++;
50       else if (e.getSource() == vansBtn) vans++;
51       else if (e.getSource() == reset)   cars = vans = 0;
52       carsTxt.setText("" + cars);
53       vansTxt.setText("" + vans);
54    }
55 }
```

Breaking up the code sections

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

public class Vehicles extends JFrame
          implements ActionListener
{
   JTextField carsTxt = new JTextField(3);
   JTextField vansTxt = new JTextField(3);
   JButton carsBtn = new JButton("Car");
   JButton vansBtn = new JButton("Van");
   JButton reset = new JButton("Reset");
   int cars = 0, vans = 0; // counters

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

the usual GUI details of buttons and text fields

two variables to hold the values for the number of car and van 'clicks'

creating the class

```
public Vehicles()
{
    setLayout(new BorderLayout());
    setSize(400, 140);
    setTitle("Car and van counter");
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    JPanel top = new JPanel();
    top.setLayout(new FlowLayout());
    top.add(new Label("Cars:"));
    top.add(carsTxt); carsTxt.setEditable(false);
    top.add(new Label("        Vans:"));
    top.add(vansTxt); vansTxt.setEditable(false);
    carsTxt.setText("0");
    vansTxt.setText("0");
    add("North", top);
    JPanel bottom = new JPanel();
    bottom.setLayout(new FlowLayout());
    bottom.add(carsBtn);
    bottom.add(vansBtn);
    bottom.add(reset);
    carsBtn.addActionListener(this);
    vansBtn.addActionListener(this);
    reset.addActionListener(this);
    add("South", bottom);
    setVisible(true);
}
```

class constructor for Vehicles setting up the content pane adding the GUI features and the listeners to the three buttons

Note the quite complex arrangements of layout to ensure the buttons, labels and text fields end up looking good

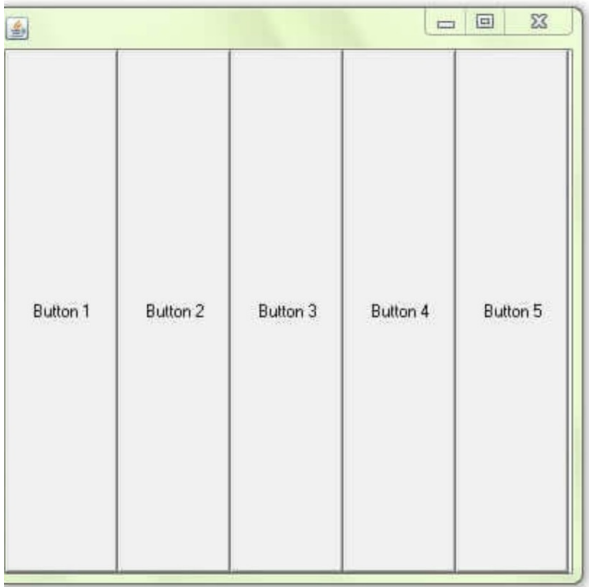
Some other Java Layout classes

- **Java GridLayout**

The **Java GridLayout** class is used to arrange the components in a rectangular grid. One component is displayed in each rectangle.

- **Java BoxLayout class**

The **Java BoxLayout class** is used to arrange the components either vertically or horizontally.



the UNIVERSITY of GREENWICH

Summary

- This section has been a very brief introduction to GUI programming using Swing.
- JFrames,JButtons, ActionListener
- JLabels, JTextFields and JTextAreas for text handling
- JCheckBoxes, JRadioButtons and JComboBoxes for making choices
- Layout manager